

# 目 录

<b>第一章 一般代数运算</b> .....	( 1 )
§1. 求一元二次方程的实根和复根.....	( 1 )
§2. 双曲函数和反双曲函数.....	( 3 )
§3. 二次方程变换 ( I ) .....	( 5 )
§4. 二次方程变换 ( II ) .....	( 6 )
§5. 二次方程变换 ( III ) .....	( 8 )
§6. 求两个整数的最大公因子.....	( 10 )
§7. 求一个整数的质因数分解.....	( 11 )
§8. 排列和组合.....	( 13 )
§9. 求A为底的对数 $\text{LOG}_A(B)$ .....	( 14 )
<b>第二章 一般几何运算</b> .....	( 16 )
§1. 矢量运算.....	( 16 )
§2. 矢量分解.....	( 18 )
§3. 直角坐标与极坐标的变换.....	( 21 )
§4. 直角坐标变换为球坐标.....	( 24 )
§5. 角度换算 ( I ) .....	( 26 )
§6. 角度换算 ( II ) .....	( 27 )
§7. 三角多项式的求值.....	( 28 )
§8. 平面三角形的解法.....	( 30 )
§9. 求平面多边形的面积.....	( 34 )
<b>第三章 基本复数运算</b> .....	( 37 )
§1. 求复数自变量的三角函数值.....	( 37 )
§2. 复数四则运算.....	( 39 )
§3. 求一个复数的平方根.....	( 42 )
§4. 复数行列式求值.....	( 43 )
§5. 列主元消去法解复线性方程组.....	( 47 )
§6. 复数矩阵的加减法.....	( 51 )
✓ §7. 复数矩阵的乘法.....	( 54 )
§8. 求复数矩阵的逆矩阵.....	( 57 )
<b>第四章 布尔代数运算</b> .....	( 63 )
§1. 求逻辑与、或、异或.....	( 63 )
§2. 两个二进制数求和.....	( 66 )
§3. 求两个二进制数的商.....	( 70 )
§4. 求两个二进制数的积.....	( 75 )
§5. 用卡诺图法化简逻辑表达式.....	( 78 )

<b>第五章 集合运算</b>	( 85 )
§1. 求两个集合的并	( 85 )
§2. 求两个集合的交	( 87 )
§3. 求两个集合的相对补	( 89 )
§4. 求两个集合的笛卡尔积	( 92 )
<b>第六章 概率计算</b>	( 95 )
§1. 计算实验数据的均值、方差并印出经验分布曲线	( 95 )
§2. 产生任意区间的均匀分布的随机数	( 99 )
§3. 产生任意均值和方差的正态分布的随机数	( 101 )
<b>第七章 特殊函数求值</b>	( 105 )
§1. 计算贝塞尔函数值	( 105 )
§2. 计算 $\Gamma$ 函数值	( 106 )
§3. 超几何函数求值	( 109 )
<b>第八章 高次代数方程求根</b>	( 111 )
§1. Newton-Raphson 法求多项式的实根	( 111 )
§2. 半区间搜索法求多项式的实根	( 113 )
§3. BH 法求多项式的实根和复根	( 116 )
<b>第九章 求定积分</b>	( 123 )
§1. 辛普生 (Simpson) 求定积分	( 123 )
§2. 高斯法求定积分	( 124 )
§3. 梯形法求定积分	( 127 )
<b>第十章 微分及龙格-库塔法求解一阶微分方程组</b>	( 130 )
§1. 微分	( 130 )
§2. 龙格-库塔法解一阶微分方程组	( 132 )
<b>第十一章 插值</b>	( 137 )
§1. 线性内插法	( 137 )
§2. 抛物线插值法	( 138 )
§3. 拉格朗日内插法	( 140 )
§4. 牛顿插值	( 142 )
§5. 三次样条函数插值	( 145 )
<b>第十二章 数据平滑与滤波</b>	( 152 )
§1. 五点三次平滑	( 152 )
§2. $\alpha$ - $\beta$ - $\gamma$ 滤波器	( 155 )
§3. 离散随机线性系统的 Kalman 滤波器	( 171 )
<b>第十三章 信号分析</b>	( 189 )
§1. 直接法进行富里叶分解	( 189 )
§2. 直接法求相关函数	( 193 )
§3. 快速富里叶分解	( 196 )
§4. 用快速富里叶变换 (FFT) 求相关函数	( 211 )
<b>第十四章 线性代数计算</b>	( 219 )

§1. 用高斯-约旦消去法求逆矩阵	( 219 )
§2. 用高斯-约旦迭代法求逆矩阵	( 223 )
§3. 矩阵相乘	( 226 )
§4. 求行列式的值	( 229 )
§5. 求实对称矩阵特征值和特征向量	( 233 )
§6. 矩阵相加、相减和标量相乘	( 239 )
§7. 矩阵求迹、求秩	( 243 )
§8. 迭代法解线性方程组	( 247 )
§9. LU分解法解线性方程组	( 250 )
§10. 高斯——约旦法解线性方程组	( 254 )
§11. 化一般矩阵为上H阵	( 258 )
§12. QR法求实上H阵特征值	( 262 )
§13. 用克雷洛夫方法求矩阵的特征多项式	( 272 )
<b>第十五章 多元线性回归分析</b>	( 277 )
<b>第十六章 一维最优化方法</b>	( 285 )
§1. 直接优选法求极值	( 285 )
§2. 二次插值法求极值	( 289 )
§3. 有理插值求极值	( 293 )
<b>第十七章 线性规划、动态规划</b>	( 298 )
§1. 非全人工变数单纯形法解线性规划问题	( 298 )
§2. 一维动态规划	( 307 )
<b>第十八章 控制系统的计算机辅助设计</b>	( 316 )
§1. 画控制系统传递函数的NYQUIST图	( 316 )
§2. 由系统的频率响应特性数据拟合系统的传递函数	( 323 )
<b>第十九章 随机服务系统的仿真与马尔可夫链分析</b>	( 337 )
§1. 随机服务系统简介	( 337 )
§2. 单队单服务员的随机服务系统仿真	( 337 )
§3. 多队单服务员的随机服务系统仿真	( 345 )
§4. 马尔可夫链分析	( 351 )
<b>第二十章 控制系统仿真</b>	( 365 )
§1. 面向微分方程的数字仿真程序	( 365 )
§2. 连续系统结构图法数字仿真程序	( 372 )
§3. 连续系统离散相似法数字仿真程序	( 384 )
§4. 单纯形法寻优程序	( 398 )
§5. 线性定常系统最佳控制仿真设计程序	( 411 )
<b>附录: BASIC语言基本语句</b>	( 426 )

# 第一章 一般代数运算

## §1. 求一元二次方程的实根和复根.

### 一、方法概要:

设二次方程为  $A_1X^2 + A_2X + A_3 = 0$

若判别式  $A_2^2 - 4A_1A_3 \geq 0$ , 二次方程有两个实根:

$$X_{1,2} = \frac{-A_2 \pm \sqrt{A_2^2 - 4A_1A_3}}{2A_1}$$

若判别式  $A_2^2 - 4A_1A_3 < 0$ , 二次方程有两个复根:

$$X_{1,2} = \frac{-A_2 \pm I \sqrt{-(A_2^2 - 4A_1A_3)}}{2A_1}$$

若判别式  $A_2^2 - 4A_1A_3 = 0$ , 二次方程有两个重根.

### 二、程序说明:

1. 程序使用工作单元  $A_1, A_2, A_3, X_1, X_2, I, D, A, T_1$
2. 程序结束, 二个根的实部在  $X_1$  和  $X_2$  中, 虚部在  $I_1$  和  $I_2$  中.

### 三、操作说明

1. 本程序由 BASIC 解释程序引入机内, 由键盘命令 RUN 启动本程序运行, 电传自动印出: "A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>" 询问二次方程的各个系数.
2. 由键盘输入二次方程系数 A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub> 之值, 机器自动印出结果:

$$X_1 = * * * + I * * *$$

$$X_2 = * * * - I * * *$$

当只有两个实根时, 只印出实部, 虚部不印出.

3. 然后机器又印出: "MORE INPUT?"  
询问还有新的二次方程要求根吗? 若有, 由键盘输入 "1", 否则回答其他任意数.
4. 若回答 "1", 机器又印出 "A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>" 重复第 (2) 步. 若回答为非 "1" 数, 程序结束.

### 四、试题:

1.  $X^2 + 2X + 3 = 0$

根:  $X_1 = -1 + \sqrt{2} I = -1 + 1.41421 I$

$X_2 = -1 - \sqrt{2} I = -1 - 1.41421 I$

2.  $X^2 + 2X + 1 = 0$

根:  $X_1 = -1$

$X_2 = -1$

3.  $12.9X^2 - 15X + 54 = 0$

根:  $X_1 = 0.6 + 1.98997 I$

$X_2 = 0.6 - 1.98997 I$



## 五、程序清单及运行结果

```
7000 PRINT "A1,A2,A3"
7002 INPUT A1,A2,A3
7004 PRINT
7006 LET T1=A2/(2*A1)
7008 LET D=A2*A2-4*A1*A3
7010 IF D<0 THEN GOTO 7020
7012 LET X1=-T1-SQR(D)/(2*A1)
7014 LET X2=-T1+SQR(D)/(2*A1)
7016 PRINT "X1="; X1,"X2=";X2
7018 GOTO 7030
7020 LET X2=-T1
7022 LET X1=-T1
7024 LET I=SQR(-D)/(2*A1)
7026 PRINT "X1=";X1,"+I";ABS(I)
7028 PRINT "X2=";X2,"-I";ABS(I)
7030 PRINT "MORE INPUT A"
7032 INPUT A
7034 PRINT
7036 IF A=1 THEN GOTO 7000
7038 END
```

运行

RUN

A1,A2,A3

? 1? 2? 3

X1=-1+I 1.41421

X2=-1-I 1.41421

MORE INPUT A?

? 1

A1,A2,A3

? 1? 2? 1

X1=-1 X2=-1

MORE INPUT A?

? 1

A1,A2,A3

? 12.5? -15? 54

X1=.6+I 1.98997

X2=.6-I 1.98997

MORE INPUT A?

? 0

## §2 双曲函数和反双曲函数

## 一、方法概要

本程序可用以计算角度自变量为 $x$ （单位：弧度）的双曲函数和反双曲函数共十个。

应注意 $x$ 的取值不应使 $e^x$ 溢出。

在本程序中将用到下列计算公式和代码：

$$\text{代码“1”代表双曲正弦} \quad \text{shx} = \sinh x = \frac{e^x - e^{-x}}{2}$$

$$\text{代码“2”代表双曲余弦} \quad \text{chx} = \cosh x = \frac{e^x + e^{-x}}{2}$$

$$\text{代码“3”代表双曲正切} \quad \text{thx} = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\text{代码“4”代表双曲余切} \quad \text{cthx} = \text{ctghx} = \frac{e^x + e^{-x}}{e^x - e^{-x}}$$

$$\text{代码“5”代表双曲正割} \quad \text{sechx} = \frac{1}{\cosh x}$$

$$\text{代码“6”代表双曲余割} \quad \text{cschx} = \frac{1}{\sinh x}$$

$$\text{代码“7”代表反双曲正弦} \quad \sinh^{-1} x = \ln(x + \sqrt{x^2 + 1})$$

$$\text{代码“8”代表反双曲余弦} \quad \cosh^{-1} x = \ln(x + \sqrt{x^2 - 1})$$

$$\text{代码“9”代表反双曲正切} \quad \tanh^{-1} x = \frac{1}{2} \ln \frac{1+x}{1-x}$$

$$\text{代码“10”代表反双曲余切} \quad \text{ctgh}^{-1} x = \frac{1}{2} \ln \frac{x+1}{x-1}$$

## 二、程序说明

程序使用工作单元N, X, V, W

N为所求函数的代码

X为自变量

## 三、操作说明

1. 由BASIC解释程序将本程序引入机内。键盘命令RUN启动本程序执行。电传印出：

"CODE, X, TO END PROGRAM INPUT 0, 0?"

2. 键盘输入所求函数的代码N及自变量X的值。电传自动印出结果并询问下次要求解的函数代码和X值。

3. 若再求新的函数，则重复2。

若不再计算，则键盘输入0启动，程序自动转到结束。

## 四 试题：

求  $\sinh(2)$ ,  $\cosh(1)$ ,  $\text{ctanh}(.5)$

ARCSINH(.5).

五、程序清单及运行结果

```
7100 PRINT "CODE,X,TO END PROGRAM INPUT 0,0"
7102 INPUT N,X
7104 PRINT
7106 IF N=0 THEN GOTO 7170
7108 IF N>6 THEN GOTO 7146
7110 LET V=(EXP(X)-EXP(-X))/2
7112 LET W=(EXP(X)+EXP(-X))/2
7114 IF N>4 THEN GOTO 7136
7116 IF N>1 THEN GOTO 7120
7118 PRINT "SINH(",X,")=";V
7119 GOTO 7166
7120 IF N>2 THEN GOTO 7126
7122 PRINT "COSH(",X,")=";W
7124 GOTO 7166
7126 IF N>3 THEN GOTO 7132
7128 PRINT "TANH(",X,")=";V/W
7130 GOTO 7166
7132 PRINT "CTANH(",X,")=";W/V
7134 GOTO 7166
7136 IF N>5 THEN GOTO 7142
7138 PRINT "SECH(",X,")=";1/W
7140 GOTO 7166
7142 PRINT "CSCH(",X,")=";1/V
7144 GOTO 7166
7146 IF N>7 THEN GOTO 7152
7148 PRINT "ARCSINH(",X,")=";LOG(X+SQR(X*X+1))
7150 GOTO 7166
7152 IF N>8 THEN GOTO 7158
7154 PRINT "ARCCOSH(",X,")=";LOG(X+SQR(X*X-1))
7156 GOTO 7166
7158 IF N>9 THEN GOTO 7164
7160 PRINT "ARCTANH(",X,")=";LOG((1+X)/(1-X))/2
7162 GOTO 7166
7164 PRINT "ARCCTANH(",X,")=";LOG((X+1)/(X-1))/2
7166 PRINT "CODE,X"
7168 GOTO 7102
7170 END
```

运行:

```

RUN
CODE,X,TO END PROGRAM INPUT 0,0
? 1? 2
SINH(2)=3.62686
CODE,X
? 2? 1
COSH(1)=1.54303
CODE,X
? 4? .5
CTANH(.5)=2.16395
CODE,X
? 7? .5
ARCSINH(.5)=.48121
CODE,X
? 0? 0
END AT 7170

```

### §3. 二次方程变换(I)

#### 一、算法概要:

本程序是将显形式的二次方程变换成普通的形式.

给定  $Y = PX + Q \pm \sqrt{RX^2 + SX + T}$

要求出:  $AX^2 + BXY + CY^2 + DX + EY + F = 0$  的形式.

其展开方法是将有理项集中在左边, 方程两边平方, 再合并同类项, 以获得要求的系数, 使之成为普通的形式.

$$Y - PX - Q = \pm \sqrt{RX^2 + SX + T}$$

$$(P^2 - R)X^2 + (-2P)XY + Y^2 + (2PQ - S)X + (-2Q)Y + (Q^2 - T) = 0$$

则  $A = P^2 - R$

$$B = -2P$$

$$C = 1$$

$$D = 2PQ - S$$

$$E = -2Q$$

$$F = Q^2 - T$$

#### 二、程序说明

本程序使用工作单元P, Q, R, S, T, A, B, C, D, E, F

计算出结果后, 部分工作单元赋值被破坏

#### 三、操作说明:

1. 由BASIC解释程序将本程序输入机内.
2. 键盘命令RUN启动本程序执行, 电传机询问P, Q, R, S, T之值.
3. 键盘输入P, Q, R, S, T的值, 电传机打印出结果A, B, C, D, E, F的值, 并询问

是否继续作题。

4. 键盘输入“1”则继续作题，重复步骤3，若不再作了，则输入“0”，程序结束。

四、试题：

$$Y=4X+5\pm\sqrt{6X^2+7X+8}$$

则      A=10      B=-8      C=1  
        D=33      E=-10      F=17

五、程序清单及运行结果

```
7300 PRINT "P,Q,R,S,T"
7302 INPUT P,Q,R,S,T
7304 PRINT
7306 PRINT "A=";P*P-R
7308 PRINT "B=";-2*P
7310 PRINT "C=";1
7312 PRINT "D=";2*P*Q-S
7314 PRINT "E=";-2*Q
7316 PRINT "F=";Q*Q-T
7318 PRINT
7320 PRINT "MORE INPUT?(1=YES, 0=NO)"
7322 INPUT P
7324 PRINT
7326 IF P=1 THEN GOTO 7300
7328 END
```

运行：

RUN

P, Q, R, S, T

? 4? 5? 6? 7? 8

A=10

B=-8

C=1

D=33

E=-10

F=17

MORE INPUT?(1=YES 0=NO)

? 0

END AT 7328

## §4. 二次方程变换 (I)

一、算法概要：

本程序是用来将一个普通的二次方程变换成显形式的二次方程。

给定:  $AX^2 + BXY + CY^2 + DX + EY + F = 0$

要求出:  $X = PY + Q \pm \sqrt{RY^2 + SY + T}$  的形式。

由一个二次解所产生的形式除以A, 得到展开式:

$$X = \left(\frac{-B}{2A}\right)Y + \left(\frac{-D}{2A}\right) \pm \sqrt{\left(\left(\frac{B}{2A}\right)^2 - \frac{C}{A}\right)Y^2 + \left(\frac{BD}{2A^2} - \frac{E}{A}\right)Y + \left(\left(\frac{D}{2A}\right)^2 - \frac{F}{A}\right)}$$

这里, 普通形式的系数排列应随所求显形式变量而异。

如: 要求  $X = PY + \dots$  用  $AX^2 + BXY + \dots$   
 $Y = PX + \dots$  则用  $AY^2 + BXY + \dots$

## 二、程序说明:

程序使用工作单元A, B, C, D, E, F, P, Q, R, S, T.

计算得出结果后, 部分工作单元初始信息破坏。

## 三、操作说明:

1. 由BASIC解释程序将程序输入机内。
2. 键盘命令RUN启动本程序执行。电传机询问A, B, C, D, E, F值。
3. 键盘输入A, B, C, D, E, F值, 电传机打印出计算结果P, Q, R, S, T值, 并询问是否继续计算。
4. 键盘输入"1", 重新计算, 重复步骤3; 输入"0", 程序结束。

## 四、试题:

将下面的普通二次方程

$$X^2 + 2XY + 3Y^2 + 4X + 5Y + 6 = 0$$

变换成显形式  $X = PY + Q \pm \sqrt{RY^2 + SY + T}$

计算结果

$$P = -1$$

$$Q = -2$$

$$R = -2$$

$$S = -1$$

$$T = -2$$

即原二次方程变换为

$$X = -Y - 2 \pm \sqrt{-2Y^2 - Y - 2}$$

## 五、程序清单及运行结果

```
7400 PRINT "ENTER A,B,C,D,E,F"
7402 INPUT A,B,C,D,E,F
7404 PRINT
7406 PRINT "P="; -B/A/2
7408 PRINT "Q="; -D/A/2
7410 PRINT "R="; (B/A/2)*(B/A/2)-C/A
7412 PRINT "S="; (B*D/2/A-E)/A
7414 PRINT "T="; (D/A/2)*(D/A/2)-F/A
7416 PRINT
7418 PRINT "MORE INPUT? (1=YES,0=NO)"
```

```

7420 INPUT A
7422 PRINT
7424 IF A=1 THEN GOTO 7400
7426 END

```

运行

RUN

ENTER A,B,C,D,E,F

? 1? 2? 3? 4? 5? 6

P=-1

O=-2

R=-2

S=-1

T=-2

MORE INPUT?(1=YES,0=NO

? 1

ENTER A,B,C,D,E,F

? 5? 6? 2? 4? 9? 8

R=-.6

Q=-.4

R=-4.00001E-2

S=.6

T=1.44

MORE INPUT?(1=YES,0=NO)

? 0

END AT 7426

## §5. 二次方程变换 (Ⅲ)

### 一、算法概要:

本程序是将显形的二次方程 $Y=f(x)$ 变换成 $X=g(y)$ 的形式

给定:  $Y=PX+Q\pm\sqrt{RX^2+SX+T}$

要求出  $X=P_1Y+Q_1\pm\sqrt{R_1Y^2+S_1Y+T_1}$  的形式。

方法是先将原始方程变换成 $Y-PX-Q=\pm\sqrt{RX^2+SX+T}$ , 等式两边平方后变换成普通形式, 然后计算出所求函数的系数。

### 二、程序说明:

程序使用工作单元P, Q, R, S, T, A, D, P<sub>1</sub>, Q<sub>1</sub>, R<sub>1</sub>, S<sub>1</sub>, T<sub>1</sub>。

计算得出结果后, 部分工作单元初始信息破坏。

### 三、操作说明:

1. 由BASIC解释程序将程序输入机内。

2. 键盘命令RUN启动本程序执行。电传机询问P, Q, R, S, T的值。

3. 键盘输入P, Q, R, S, T值后, 电传机打印出结果 $P_1$ ,  $Q_1$ ,  $R_1$ ,  $S_1$ 和 $T_1$ 值, 并询问是否继续作题。

4. 欲继续作题, 键盘输入"1", 如不再作, 键盘输入"0", 程序结束。

#### 四、试题:

$Y=2X+5 \pm \sqrt{8X^2+9X+4}$ , 变换成一个Y函数,

即 $X=P_1Y+Q_1 \pm \sqrt{R_1Y^2+S_1Y+T_1}$ 形式。

计算结果

$$P_1 = \frac{P}{A} = -0.5$$

$$Q_1 = -\frac{D}{2A} = 1.375$$

$$R_1 = \left(\frac{P}{A}\right)^2 - \frac{1}{A} = 0.5$$

$$S_1 = -DP/A^2 + 2Q/A = -3.875$$

$$T_1 = \left(\frac{D}{2A}\right)^2 - (Q^2 - T)/A = 7.141$$

其中  $A = P * P - R$ ,  $D = 2 * P * Q - S$

#### 五、程序清单及运行结果

```

7400 PRINT "ENTER P,Q,R,S,T"
7404 INPUT P,Q,R,S,T
7406 PRINT
7408 LET A=P * P-R
7410 LET D=2 * P * Q-S
7412 PRINT "P1=";P/A
7414 PRINT "Q1=";-D/2/A
7416 PRINT "R1=";(P/A) * (P/A)-1/A
7418 PRINT "S1=";-D * P/(A * A)+2 * Q/A
7420 PRINT "T1=";(D/2/A) * (D/2/A)-(Q * Q-T)/A
7422 PRINT
7424 PRINT "MORE INPUT?(1=YES,0=NO)"
7426 INPUT P
7428 PRINT
7430 IF P=1 THEN GOTO 7400
7432 END

```

运行:

RUN

ENTER P,Q,R,S,T

? 2? 5? 8? 9? 4

P1=-.5

Q1=1.375

R1=.5



```

S1=-3.875
T1=7.14062
MORE INPUT?(1=YES,0=NO)
? 0
END AT 7432

```

## §6 求两个整数的最大公因子

### 一、方法概要

本程序用欧几里德算法计算两个整数的最大公因子。

$x, y$  为两个整数, 令

$$C=|x|, \quad D=|y|$$

则  $R=C-D*(C/D \text{ 的整数部分})$  (1-6-1)

若  $R=0$ , 则  $D$  即为  $x$  和  $y$  的最大公因子。若  $R \neq 0$ , 再令  $C=D, D=R$ , 重新按公式(1-6-1) 计算出新的  $R$  值, 周而复始, 直至  $R=0$  找到最大公因子。

### 二、程序说明

本程序使用工作单元  $A, B, C, D, R$ 。运算结果存在  $D$  中。

### 三、操作说明

1. 由BASIC解释程序将本程序输入机内。
2. 键盘命令RUN启动本程序执行, 电传机询问两个整数的值?
3. 键盘输入两个整数, 电传机立即打出结果:

$G.C.D.=$

并询问另两个整数。

4. 如继续计算, 则重复步骤 3, 如不再计算, 则输入 0, 程序结束。

### 四、试题:

求45和285的最大公因子。

计算结果  $G.C.D.=15$

### 五、程序清单和运行结果

```

7500 PRINT "GREATEST COMMON DIVISOR OF TWO INTEGERS"
7502 PRINT
7504 PRINT "INPUT INTEGER, INTEGER TO END PROGRAM INPUT 0,0"
7506 INPUT A,B
7508 PRINT
7510 IF A=0 THEN GOTO 7536
7512 PRINT
7514 LET C=ABS(A)
7516 LET D=ABS(B)
7518 LET R=C-D*INT(C/D)
7520 IF R=0 THEN GOTO 7528
7522 LET C=D

```

```

7524 LET D=R
7526 GOTO 7518
7528 PRINT "G.C.D=",D
7530 PRINT
7532 PRINT "INPUT INTEGER, INTEGER"
7534 GOTO 7506
7536 END
运行:
RUN
GREATEST COMMON DIVTSOR OF TWO INTEGERS
INPUT INTEGER, INTEGER. TO END PROGRAM INPUT 0,0
? 45? 285
G.C.D=15
INPUT INTEGER, INTEGER
? 57? 33
G.C.D= 3
INPUT INTEGER, INTEGER
? 57? 11
G.C.D=1
INPUT INTEGER, INTEGER
? 0? 0
END AT 7536

```

## §7 求一个整数的质因数分解

### 一、程序说明:

1. 本程序用来将一个整数分解成质数的乘积, 使该整数

$$N = (A_1)^{B_1} * (A_2)^{B_2} * \dots * (A_N)^{B_N}$$

2. 程序占用工作单元N, I, S。

### 二、操作说明:

1. 由BASIC解释程序将本程序输入机内
2. 键盘命令RUN启动本程序执行, 电传机询问所要分解的整数N值
3. 键盘输入N值, 电传机打印出计算结果:

1 (或-1, 表示数的正负号)

$A_1$  ↑  $B_1$  (↑表示乘幂运算符)

$A_2$  ↑  $B_2$

⋮ ⋮ ⋮

$A_N$  ↑  $B_N$

当输入数为0, 1或非整数时, 不予回答。

电传机印出: "NUMBER?", 重新询问N值, 返回3。

三、试题:

$$N=84=7 \times 3 \times (2)^2 \times 1$$

四、程序清单及运行结果

```
7800 PRINT "INPUT NUMBER TO BE FACTORED,TO END PROGRAM IN-
      PUT 0"
7802 INPUT N
7804 PRINT
7806 IF N=0 THEN GOTO 7840
7808 PRINT
7810 PRINT "FACTORS:"
7812 PRINT SGN(N)
7814 LET N=ABS(N)
7816 FOR I=2 TO N
7818     LET S=0
7820     IF N/I>INT(N/I) THEN GOTO 7828
7822     LET N=N/I
7824     LET S=S+1
7826     GOTO 7820
7828     IF S=0 THEN GOTO 7832
7830     PRINT I, "↑"; S
7832 NEXT I
7834 PRINT
7836 PRINT "NUMBER";
7838 GOTO 7802
7840 END
```

运行:

RUN

INPUT NUMBER TO BE FACTORED,TO END PROGRAM INPUT 0

? 84

FACTORS,

1

2 ↑ 2

3 ↑ 1

7 ↑ 1

NUMBER? 64

FACTORS,

1

2 ↑ 6

NUMBER? 158

```
FACTORS,
1
2 ↑ 1
79 ↑ 1
NUMBER? 0
END AT 7840
```

## §8 排列和组合

### 一、算法概要:

本程序用以计算从N项中每次取R项的排列和组合值。

所用公式: N项中每次取R的

$$\text{排列值} = \frac{N!}{(N-R)!} = N(N-1)(N-2)\cdots(N-R+1)$$

$$\text{组合值} = \frac{N!}{R!(N-R)!} = \frac{\text{排列值}}{R!}$$

其中“!”是阶乘, 例如:  $5! = 5 * 4 * 3 * 2 * 1$

### 二、程序说明:

程序占用工作单元N, R, P, L, F, I

### 三、操作说明:

1. 由BASIC解释程序将本程序输入机内。
2. 键盘命令RUN↵, 启动本程序执行。电传机询问N, R值。
3. 键盘输入N, R值, 电传机打印出结果并询问新的N, R值。若结果的值太大, 则算术溢出, 给出出错信息 POINT OVERFLOW
4. 若继续计算则重复步骤3, 如不再计算则输入等于或小于0的N, R值, 程序结束。

### 四、试题:

1. 6项中每次取4项的排列值=360  
6项中每次取4项的组合值=15
2. 15项中每次取2项的排列值=210  
15项中每次取2项的组合值=105

### 五、程序清单和运行结果

```
7900 PRINT "INPUT N, R (TO END PROGRAM INPUT 0, 0)"
7902 GOTO 7806
7904 PRINT "INPUT N,R"
7906 INPUT N,R
7908 PRINT
7910 IF N=<0 THEN GOTO 7958
7912 IF R=<0 THEN GOTO 7958
7914 IF R=<N THEN GOTO 7922
```

```

7916 PRINT " MUST BE R<=N"
7918 PRINT
7920 GOTO 7904
7922 LET P=1
7924 FOR L=N-R+1 TO N
7926 IF 1.70141E+38/L=>P THEN GOTO 7936
7930 PRINT" POINT OVERFLOW"
7932 PRINT
7934 GOTO 7904
7936 LET P=P*L
7938 NEXT L
7940 LET F=1
7942 IF R=1 THEN GOTO 7950
7944 FOR I=2 TO R
7946 LET F=F*I
7948 NEXT I
7950 PRINT "SUM OF PERMUTATIONS IS", P
7952 PRINT "SUM OF COMBINATIONS IS", P/F
7954 PRINT
7956 GOTO 7904
7958 END

```

运行

RUN

INPUT N, R (TO END PROGRAM INPUT 0, 0)

? 6 ? 4

SUM OF PERMUTATIONS IS 360

SUM OF COMBINATIONS IS 15

INPUT N, R

? 15 ? 2

SUM OF PERMUTATIONS IS 210

SUM OF COMBINATIONS IS 105

INPUT N, R

? 0 ? 0

END AT 7958

## §9 求A为底的对数 $\text{LOG}_A(B)$

一、算法概要:

用  $\text{LOG}_A(B) = \frac{\text{LOG}_e(B)}{\text{LOG}_e(A)}$  计算A为底的对数 $\text{LOG}_A(B)$

本程序适用于求以任意正数A为底的数B的对数。

## 二、程序说明:

本程序使用工作单元A, B。

## 三、操作说明:

1. 由BASIC解释程序将本程序输入机内
2. 键盘命令RUN↵, 启动本程序执行。电传机询问A、B值
3. 键盘输入A、B值, 电传机打印出计算结果, 并询问新的A、B值。
4. 如需继续计算, 则重复步骤3, 如不再计算, 则输入0, 0, 程序结束。

## 四、试题:

1.  $\text{LOG}_{2.5}(3)$   
 $\text{LOG } 3 \text{ TO BASE } 2.5 = 1.19898$
2.  $\text{LOG}_{2.732}(10)$   
 $\text{LOG } 10 \text{ TO BASE } 2.732 = 2.29105$   
当 $A \leq 0$ 时, 不予回答, 程序结束

## 五、程序清单及运行结果:

```
7200 PRINT "INPUT A, B, TO END PROGRAM INPUT 0,0"
7202 INPUT A,B
7204 PRINT
7206 IF A<= 0 THEN GOTO 7216
7208 PRINT "LOG"; B; "TO BASE" ..A; "=", LOG(B)/LOG(A)
7210 PRINT
7212 PRINT "INPUT A,B"
7214 GOTO 7202
7216 END
```

### 运行结果

RUN

INPUT A, B TO END PROGRAM INPUT 0, 0

? 2.5? 3

LOG 3 TO BASE 2.5=1.19898

INPUT A, B

? 10? 10

LOG 10 TO BASE 10=1

INPUT A, B

? 2.732 ? 10

LOG 10 TO BASE 2.732=2.29105

INPUT A, B

? 0 ? 0

END AT 7216

## 第二章 一般几何运算

### §1 矢 量 运 算

#### 一、方法概要

本程序完成三维空间中二个矢量间的矢量加法, 矢量减法, 点乘以及叉积。

已知两个矢量:

$$A = (A(1), A(2), A(3))$$

$$B = (B(1), B(2), B(3))$$

矢量相加:

$$A + B = (A(1) + B(1), A(2) + B(2), A(3) + B(3))$$

$$A - B = (A(1) - B(1), A(2) - B(2), A(3) - B(3))$$

$$A \cdot B = (A(1)B(1) + A(2)B(2) + A(3)B(3))$$

$$A \times B = (A(2)B(3) - B(2)A(3), A(3)B(1) - B(3)A(1), \\ A(1)B(2) - A(2)B(1))$$

#### 二、程序说明

(一) 本程序使用的工作单元

$A(3)$ ,  $B(3)$  是矢量运算的两个三维矢量,

其他都作为中间变量, 包括  $A$ ,  $B$ ,  $C$ ,  $D$ ,  $E$

(二) 本程序可用作子程序, 当作为子程序使用时, 如果  $A(I)$ ,  $B(I)$  都已在主程序中, 则可不要输入部分, 直接从 6914 语句进入本程序, 同时, 在结束处需将 END 改为 RETURN。

#### 三、操作说明

1. 将该程序送入机内。
2. 由键盘命令 RUN  $\swarrow$  启动本程序执行, 电传机印出:  
"INPUT VECTOR A"
3. 由键盘输入矢量  $A$ , 输完后电传机印出:  
"INPUT VECTOR B"
4. 由键盘输入矢量  $B$ , 机器便开始运算, 电传机印出:  
"OUTPUT VECTORS", 然后打出全部计算结果。
5. 计算结束后, 电传机打出 "MORE INPUT (1=YES, 0=NO)"  
用户必须作出回答: 回答 "1" 则本程序重新启动, 若回答 "0", 则程序停止。

#### 四、试题:

$$A = (1, 5, 7)$$

$$B = (2, -3, 4)$$

打印结果:

$$A + B = (3, 2, 11)$$

$$A - B = (-1, 8, 3)$$

$$A \cdot B = 15$$

$$A \times B = (41, 10, -13)$$

##### 五、程序清单及运行结果

```

6900 REM VECTOR COMPUTING
6901 DIM A (3), B(3)
6902 PRINT "INPUT VECTOR A"
6904 INPUT A(1), A(2), A(3)
6906 PRINT
6908 PRINT "INPUT VECTOR B"
6910 INPUT B(1), B(2), B(3)
6912 PRINT
6914 PRINT "OUTPUT VECTORS"
6916 PRINT "A+B=( "; A(1)+B(1); ", "; A(2)+B(2); ", "; A(3)+B(3); " )"
6918 PRINT "A-B=( "; A(1)-B(1); ", "; A(2)-B(2); ", "; A(3)-B(3); " )"
6920 PRINT "A · B="; A(1) * B(1)+A(2) * B(2)+A(3) * B(3)
6922 LET C=A(2) * B(3)-A(3) * B(2)
6924 LET D=A(3) * B(1)-A(1) * B(3)
6926 LET E=A(1) * B(2)-A(2) * B(1)
6928 PRINT "A × B=( "; C; ", "; D; ", "; E; " )"
6930 PRINT
6932 PRINT "MORE INPUT (1=YES, 0=NO)"
6934 INPUT A
6936 PRINT
6938 IF A=1 THEN GOTO 6902
6940 END

```

运行:

RUN

INPUT VECTOR A

? 1 ? 5 ? 7

INPUT VECTOR B

? 2 ? - 3 ? 4

OUTPUT VECTORS

A+B=( 3 , 2 , 11)

A-B=(-1, 8, 3)

A · B=15

A × B=(41, 10, -13)

MORE INPUT (1=YES, 0=NO)

? 0

END AT 6940



## §2. 矢量分解

### 一、计算方法概要

计算三维矢量的大小(模)、矢量与轴线之间的夹角、以及两矢量之间的夹角。

模:  $M = \sqrt{X^2 + Y^2 + Z^2}$

矢量与X轴间的夹角  $= \cos^{-1}(X / \sqrt{X^2 + Y^2 + Z^2})$

矢量与Y轴间的夹角  $= \cos^{-1}(Y / \sqrt{X^2 + Y^2 + Z^2})$

矢量与Z轴间的夹角  $= \cos^{-1}(Z / \sqrt{X^2 + Y^2 + Z^2})$

两矢量之间的夹角:

$$\cos^{-1}\left(\frac{X_1 X_2 + Y_1 Y_2 + Z_1 Z_2}{\sqrt{X_1^2 + Y_1^2 + Z_1^2} \cdot \sqrt{X_2^2 + Y_2^2 + Z_2^2}}\right)$$

### 二、程序说明:

#### (一) 本程序所用的工作单元

$X(3 * N)$ , 输入的各个矢量元素, 因为本程序是分解三维空间矢量, 所以每个矢量有三个元素, 分别表示X, Y, Z坐标值。

$M(N)$ , 每个矢量的模

$A(I)$ , 每个矢量分别与X, Y, Z的坐标轴的夹角,  $I=1 \sim 3$

N, 输入要求分解的矢量个数。

(二) 本程序也可作为子程序使用, 在作为子程序使用时, 如果在主程序里已有要计算的各矢量, 则可将输入部分不用, 直接从6614语句进入本子程序, 同时, 将本程序结尾的END语句改成RETURN。

### 三、操作说明

1. 用BASIC解释程序将本程序送入机内
2. 用键盘命令RUN使本程序投入运行。
3. 电传机打印出:

“INPUT NUMBER OF VECTORS”

这时用户根据要投入计算的一组矢量, 用矢量的个数进行回答(即为N赋值)

4. 回答完后, 电传机又打印出:

“INPUT VECTOR ELEMENT LINE X, Y, Z”

这时用户输入每一个矢量中的各个元素, 输入次序, 按X, Y, Z坐标输入, 直到把全部矢量的全部元素输入进去。

5. 输入结束后, 计算机开始计算, 算出每个矢量的模, 以及各个矢量分别同X, Y, Z三个坐标轴的夹角。并在电传机上打印出计算结果。
6. 电传打印出问话

“ANGLE BETWEEN VECTOR  $J_1$  AND VECTOR  $J_2$ ”向用户询问, 求哪两个矢量之间的夹角。 $J_1$ 为第一个矢量,  $J_2$ 为第二个矢量, 用户选择两个矢量, 并将这两个矢量的序号输入到计算机内, 如用户要计算矢量1和矢量3之间的夹角, 回答1和3。求出结果后机器再次询问  $J_1, J_2$  即可求另外二个矢量的夹角, 若不再求二个矢量间的夹角, 只需对  $J_1$  输入0, 则程序转去询问是否还要重新输入矢量。

7. 电传打印出问话: "MORE INPUT(1=YES, 0=NO)"

回答 "1" 表示要再运行本程序, 如果不再计算, 则回答 "0", 程序结束。

输出的两个向量间的夹角及各向量与X, Y, Z轴间的夹角单位为 "度"。

#### 四、试题:

矢量1, X, Y, Z三个分量为 1, -4, 2

矢量2, X, Y, Z三个分量为 7, -5, -6

矢量3, X, Y, Z三个分量为 3, 5, 8

以矢量1为例, 其计算结果为:

模MAGNITUDE=4.582

与X轴的夹角 77.4°

与Y轴的夹角 -29°

与Z轴的夹角 64°

#### 五、程序清单及运行结果

```
6592 REM "PROGRAMMING OF RESOLUING VECTOR"
6594 PRINT "INPUT NUMBER OF VECTORS"
6596 INPUT N
6598 DIM X(3 * N), M(N), A(3)
6600 PRINT
6602 PRINT "INPUT VECTOR ELEMENT LINE X,Y,Z,"
6604 FOR I=1 TO N * 3
6606     INPUT X(I)
6608     IF I/3 < > INT(I/3) GOTO 6611
6610     PRINT
6611 NEXT I
6612 PRINT
6613 LET I=0
6614 FOR J=1 TO 3 * N STEP 3
6615     LET I=I+1
6616     LET M(I)= SQR(X(J) * X(J)+X(J+1) * X(J+1)+X(J+2) * X(J+2))
6618     IF M(I)= 0 GOTO 6640
6620     PRINT "FOR VECTOR=";I
6622     PRINT "MAGNITUDE=";M(I)
6624     FOR K=1 TO 3
6626         LET B=X(J+K-1)/M(I)
6628         LET A(K)= ATN (SQR(1-B*B)/B) * 57.2958
6630     NEXT K
6632     PRINT "ANGLE BETWEEN VECTOR AND X-AXIS=";A(1)
6634     PRINT "ANGLE BETWEEN VECTOR AND Y-AXIS=";A(2)
6636     PRINT "ANGLE BETWEEN VECTOR AND Z-AXIS=";A(3)
6638     PRINT
```

```

6640 NEXT J
6642 LET B=0
6644 PRINT "ANGLE BETWEEN VECTOR J1 AND VECTOR J2"
6645 INPUT J1, J2
6646 PRINT
6648 IF J1=0 GOTO 6670
6650 IF M(J1)=0 GOTO 6669
6652 IF M(J2)=0 GOTO 6669
6653 LET I1=(J1-1)*3
6654 LET I2=(J2-1)*3
6655 LET B=X(I1+1)*X(I2+1)+X(I1+2)*X(I2+2)+X(I1+3)*X(I2+3)
6656 LET B=B/(M(J1)*M(J2))
6658 IF B<>0 GOTO 6664
6660 LET B=90
6662 GOTO 6666
6664 LET B=ATN(SQR(1-B*B)/B)*57.2958
6666 PRINT"ANGLE BETWEEN VECTOR AND VECTOR=",B
6668 PRINT
6669 GOTO 6642
6670 PRINT "MORE INPUT(1=YES, 0 =NO)"
6672 INPUT B
6674 IF B=1 GOTO 6600
6676 END

```

RUN

INPUT NUMBER OF VECTORS

? 3

INPUT VECTOR ELEMENT LINE X,Y,Z

? 1? -4? 2

? 7? -5? -6

? 3? 5? 8

FOR VECTOR 1

MAGNITUDE=4.58257

ANGLE BETWEEN VECTOR AND X-AXIS=77.3956

ANGLE BETWEEN VECTOR AND Y-AXIS=-29.2059

ANGLE BETWEEN VECTOR AND Z-AXIS=64.1233

FOR VECTOR 2

MAGNITUDE=10.4881

ANGLE BETWEEN VECTOR AND X-AXIS=48.1314

```

ANGLE BETWEEN VECTOR AND Y-AXIS=-61.5278
ANGLE BETWEEN VECTOR AND Z-AXIS=-55.1048
FOR VECTOR 3
MAGNITUDE=9.89949
ANGLE BETWEEN VECTOR AND X-AXIS=72.3594
ANGLE BETWEEN VECTOR AND Y-AXIS=59.6636
ANGLE BETWEEN VECTOR AND Z-AXIS=36.0871

ANGLE BETWEEN VECTOR J1 AND VECTOR J2
? 1? 2
ANGLE BETWEEN VECTOR AND VECTOR=71.8145

ANGLE BETWEEN VECTOR J1 AND VECTOR J2
? 0? 0
MORE INPUT(1=YES, 0=NO)
? 0
END AT 6676

```

### §3 直角坐标与极坐标的变换

#### 一、方法概要

本程序可将笛卡尔坐标换成极坐标，亦可将极坐标换成笛卡尔坐标。

本程序由  $R = \sqrt{X^2 + Y^2}$

$$A = \text{tg}^{-1}(Y/X)$$

将笛卡尔坐标 (X, Y) 变换成极坐标 (R, A)。

$$\text{由 } X = R * \text{COS}(A)$$

$$Y = R * \text{SIN}(A)$$

将极坐标 (R, A) 变换成笛卡尔坐标 (X, Y)。

求出的角度范围：0—360°

#### 二、程序说明

1. 程序使用数组X, Y, R, A, Z及工作单元N, M, I等。
2. 程序结束, X, Y工作单元数值破坏。结果直接输出, 没有保存。
3. 输入输出的角度值均以“度”为单位。

#### 三、操作说明

1. 由BASIC解释程序将本程序输入机内。
2. 键盘输入命令RUN启动本程序执行。电传印出：“INPUT N, M”询问问题类别N及所要转换的点数M。

若 N=1, 由笛卡尔坐标变换成极坐标。

N=-1, 由极坐标变换成笛卡尔坐标。

M——要变换的点的个数 (不得超过20)

3. 键盘输入N, M值。电传自动印出:"INPUT X, Y" (或"INPUT R, A")
4. 键盘输入X, Y (或R, A) 值, 电传立即印出转换结果。然后继续询问N、M并重复3, 4直至回答0, 0程序结束。
5. 若R、A值 $\leq 0$ , 此点不计, 重新输入该点的值。

#### 四、试题

N=1 M=3

计算结果:

X <sub>(1)</sub> =6	}	R=8.4853	A=45
Y <sub>(1)</sub> =6			
X <sub>(2)</sub> =-6	}	R=8.4853	A=135
Y <sub>(2)</sub> =6			
X <sub>(3)</sub> =0	}	R=7	A=270
Y <sub>(3)</sub> =-7			

#### 五、程序清单及运行结果

```

7100 DIM X(20),Y(20),R(20),A(20),Z(20)
7102 PRINT "INPUT N,M"
7104 INPUT N,M
7106 PRINT
7108 IF N=0 THEN GOTO 7232
7110 IF N=-1 THEN GOTO 7180
7112 PRINT "INPUT X, Y"
7114 FOR I=1 TO M
7116 INPUT X(I), Y(I)
7118 NEXT I
7120 PRINT
7122 PRINT "X","Y","R","A"
7124 FOR I=1 TO M
7126 IF X(I)=0 THEN GOTO 7150
7128 IF Y(I)=0 THEN GOTO 7168
7130 PRINT X(I),Y(I),SQR(X(I)*X(I)+Y(I)*Y(I)),
7132 IF X(I)<0 THEN GOTO 7146
7134 IF Y(I)<0 THEN GOTO 7142
7136 PRINT ATN(Y(I)/X(I))*180/3.14159
7138 NEXT I
7140 GOTO 7230
7142 PRINT 360+ATN(Y(I)/X(I))*180/3.14159
7144 GOTO 7138
7146 PRINT 180+ATN(Y(I)/X(I))*180/3.14159
7148 GOTO 7138

```

```

7150 IF Y(I)=0 THEN GOTO 7164
7152 PRINT X(I),Y(I),ABS(Y(I)),
7154 IF Y(I)<0 THEN GOTO 7160
7156 PRINT 90
7158 GOTO 7138
7160 PRINT 270
7162 GOTO 7138
7164 PRINT X(I),Y(I),0,0
7166 GOTO 7138
7168 PRINT X(I),Y(I),ABS(X(I)),
7170 IF X(I)<0 THEN GOTO 7176
7172 PRINT 0
7174 GOTO 7138
7176 PRINT 180
7178 GOTO 7138
7180 PRINT "INPUT R,A"
7182 FOR I=1 TO M
7184   INPUT R(I),A(I)
7186   IF R(I)<0 THEN GOTO 7190
7188   IF A(I)=>0 THEN GOTO 7194
7190   PRINT "R,A<0 NO SOLUTION"
7192 GOTO 7182
7194 NEXT I
7196 PRINT
7198 PRINT "R","A","X","Y"
1200 FOR I=1 TO M
7202   LET Z(I)=(A(I)-INT(A(I)/360)*360)*3.14159/180
7204   PRINT R(I),A(I),
7206   IF Z(I)*180/3.14159>90 THEN GOTO 7214
208   PRINT R(I)*COS(Z(I)),R(I)*SIN(Z(I))
7210 NEXT I
7212 GOTO 7230
7214 IF Z(I)*180/3.14159>180 THEN GOTO 7220
7216 PRINT -R(I)*SIN(Z(I)-1.5708),R(I)*COS(Z(I)-1.5708)
7218 GOTO 7210
7220 IF Z(I)*180/3.14159>270 THEN GOTO 7226
7222 PRINT -R(I)*COS(Z(I)-3.14159),-R(I)*SIN(Z(I)-3.14159)
7224 GOTO 7210
7226 PRINT R(I)*SIN(6.28319-Z(I)), -R(I)*COS(6.28318-Z(I))
7228 GOTO 7210

```

7230 GOTO 7102

7232 END

运行结果

RUN

INPUT N,M

? 1 ? 3

INPUT X,Y

? 6 ? 6 ? -6 ? 6 ? 0 ? -7

X	Y	R	A
6	6	8.48528	45.0001
-6	6	8.48528	135
0	-7	7	270

INPUT N,M

? -1 ? 3

INPUT R,A

? 5 ? 90 ? 8.48528 ? 135 ? 7 ? 270

R	A	X	Y
5	90	2.38419E-5	5
8.48528	135	-5.99997	6.00003
7	270	-1.04862E-5	-7

INPUT N,M

? 0 ? 0

END AT 7232

## §4 直角坐标变换为球坐标

### 一、方法概要

本程序使用下列公式将直角三维空间坐标变换为球坐标:

$$\text{幅角 } \beta = \left(1 - \frac{X}{|X|}\right) \frac{\pi}{2} + \left(1 + \frac{X}{|X|}\right) \frac{\pi}{2} \left(1 - \frac{X}{|Y|}\right) + \left(\frac{X}{|X|} / \frac{Y}{|Y|}\right) \sin^{-1} \frac{|Y|}{\sqrt{X^2 + Y^2}}$$

$$\text{仰角 } \varepsilon = \sin^{-1} Z/R$$

$$\text{模 } R = \sqrt{X^2 + Y^2 + Z^2}$$

### 二、程序说明:

1. 程序使用数组X, Y, Z及工作单元M, Q, I, P, X<sub>1</sub>, Y<sub>1</sub>, B, R, E
2. 每计算一点的变换值需立即印出结果, 每次P, X<sub>1</sub>, Y<sub>1</sub>, B, R, E的值都将被破坏, 而由所求下一点的变换值所替代。
3. M——欲求变换坐标的点的个数。
4. 输出的角度值以“弧度”为单位。

### 三、操作说明

1. 由BASIC解释程序将本程序输入机内

2. 键盘命令RUN启动本程序执行,机器询问要转换坐标的点数M
3. 键盘输入M值(不得超过20),机器询问M个点每个点的直角坐标X, Y, Z值
4. 顺序输入各点的X, Y, Z值,机器立即将其转换成该点球坐标的模R,幅角 $\beta$ ,仰角 $\varepsilon$ 的值,并打印出结果。
5. 机器又询问需要转换坐标的新的点的个数M,若回答新的个数不为零,则重复步骤3, 4. 若回答0, 则程序结束。

#### 四、试题:

M=3

点1     X=2 ,     Y=5 ,     Z=9

点2     X=-2,     Y=-5,     Z=-9

点3     X=8 ,     Y=-6,     Z=-3

计算结果:      $\beta_1=3.075$       $\varepsilon_1=1.032$       $R_1=10.48$   
                    $\beta_2=4.332$       $\varepsilon_2=-1.032$       $R_2=10.48$   
                    $\beta_3=-1.69$       $\varepsilon_3=-0.29$       $R_3=10.44$

#### 五、程序清单和运行结果

```

7300 DIM X(20),Y(20),Z(20)
7302 DEF FNS(Q)=ATN(Q/SQR(1-Q*Q))
7304 PRINT "INPUT M (TO END PROGRAM INPUT 0)"
7306 INPUT M
7308 PRINT
7310 IF M=0 THEN GOTO 7354
7312 PRINT "INPUT X, Y, Z"
7314 FOR I=1 TO M
7316   INPUT X(I),Y(I),Z(I)
7318   PRINT
7320 NEXT I
7322 PRINT
7324 PRINT "X";TAB(8);"Y";TAB(16);"Z";TAB(24);"BATA";
7326 PRINT TAB(36);"EPSEL";TAB(48);"R"
7328 FOR I=1 TO M
7330   LET P=SQR(X(I)*X(I)+Y(I)*Y(I))
7332   LET X1=ABS(X(I))
7334   LET Y1=ABS(Y(I))
7336   LET B=3.1416*((1-X(I)/X1)+(1+X(I)/X1)*(1-X(I)/Y1))/2
7338   LET B=B+(X(I)*Y1/Y(I)/X1)*FNS(Y1/P)
7340   LET R=SQR(X(I)*X(I)+Y(I)*Y(I)+Z(I)*Z(I))
7342   LET E=FNS(Z(I)/R)
7344   PRINT X(I);TAB(8);Y(I);TAB(16);Z(I);
7346   PRINT TAB(24);B;TAB(36);E;TAB(48);R

```



```

7348 NEXT I
7350 PRINT "INPUT M"
7352 GOTO 7306
7354 END

```

运行:

RUN

INPUT M (TO END PROGRAM INPUT 0)

? 3

INPUT X, Y, Z

? 2? 5? 9

? -2? -5? -9

? 8? -6? -3

X	Y	Z	BATA	EPSEL	R
2	5	9	3.07525	1.03159	10.4881
-2	-5	-9	4.33189	-1.03159	10.4881
8	-6	-3	-1.6907	-0.291457	10.4403

INPUT M

? 0

END AT 7354

## §5 角度换算 (I)

### 一、方法概要:

本程序是将一个给定为弧度的角度换算成度、分、秒。公式:

$$1 \text{ 弧度} = \frac{180^\circ}{\pi}$$

换算后得到0—360°之间的一个角度的度、分、秒。

### 二、程序说明:

程序使用工作单元X, A, D, D<sub>1</sub>, M。

X 为给定角的弧度数。

### 三、操作说明:

1. 由BASIC解释程序将本程序输入机内
2. 键盘命令 RUN ↵ 启动本程序执行, 电传机询问要变换的角度值, 印出: "ANGLE?  
(TO END PROGRAM INPUT 99999)"
3. 键盘输入角度X值, 电传机印出结果  
X值          RADIANS=\*\*\*DEG\*\*\*MIN\*\*\*SEC  
并询问下次需计算的角度: ANGLE?
4. 如不再计算, 由键盘输入99999, 程序结束。

### 四、例题:

X=1          radians=57 deg 17 min 45 sec

X=6.728 radians=25 deg 29 min 10.5 sec

## 五、程序清单及运行结果

```

7540 PRINT "ANGLE(TO END PROGRAM INPUT 99999)"
7542 INPUT X
7544 PRINT
7546 IF X=99999 THEN GOTO 7562
7548 LET A=3600 * 180 * X/3.14159
7550 LET D=INT(A/3600)
7552 LET D1=INT(D/360)
7554 LET M=INT((A-D * 3600)/60)
7556 PRINT X;"RADIANS=";D-360 * D1;"DEG";M;"MIN";A-D * 3600-M * 60;"SEC"
7558 PRINT "ANGLE"
7560 GOTO 7542
7562 PRINT
7564 END

```

## 运行结果

```

RUN
ANGLE(TO END PROGRAM INPUT 99999)
? 1
1 RADIANS=57 DEG 17 MIN 45 SEC
ANGLE
? 6.728
6.728 RADIANS=25 DEG 29 MIN 11 SEC
ANGLE
? 99999
END AT 7564

```

## §6 角 度 换 算 (II)

### 一、方法概要:

本程序是将形式为度、分、秒的角度换算成弧度。公式:

$$1 \text{ 弧度} = \frac{180}{\pi}$$

### 二、程序说明:

程序占用工作单元D, S, M, A, R。

D, S, M为给定角的度、分、秒数。

### 三、操作说明:

1. 由BASIC解释程序将本程序输入机内。
2. 键盘命令RUN, 启动本程序执行, 电传机询问给定角的度、分、秒数, 键出。

"ANGLE (DEG, MIN, SEC) "

3. 键盘输入度、分、秒之值，电传机打出结果：

ANGLE =  $\times \times \times$  RADIANS

并印出："MORE INPUT (1=YES, 0=NO) "

询问是否要继续换算。

4. 键盘输入 "1" 则继续计算，输入 "0"，则程序结束。

四、例题：

$30^{\circ}5'27'' = 0.525184$  弧度。

五、程序清单及运行结果

```
7570 PRINT
7572 PRINT "ANGLE (DEG, MIN, SEC) "
7574 INPUT D, M, S
7576 PRINT
7578 LET A=D+M/60+S/3600
7580 LET R=INT (A/360)
7582 PRINT "ANGLE= "; A*1.74533E-2 - R*6.28318; "RADIANS"
7584 PRINT
7586 PRINT "MORE INPUT (1=YES, 0=NO) "
7588 INPUT D
7590 PRINT
7592 IF D=1 THEN GOTO 7570
7594 END
```

运行结果

RUN

ANGLE (DEG, MIN, SEC)

? 30 ? 5 ? 27

ANGLE=.525184 RADIANS

MORE INPUT (1=YES, 0=NO)

? 1

ANGLE (DEG, MIN, SEC)

? 57 ? 17 ? 45

ANGLE=.1 RADIANS

MORE INPUT (1=YES, 0=NO)

? 0

END AT 7594

## §7 三角多项式求值

一、方法概要：

本程序用以计算下列形式的三角多项式的值：

$$F(X) = A_1 \sin(X) + B_1 \cos(X) + A_2 \sin(2X) + B_2 \cos(2X) + \dots + A_N \sin(NX) + B_N \cos(NX)$$

在计算中，使用下面的恒等式：

$$\sin((N+1)X) = \sin(NX)\cos(X) + \cos(NX)\sin(X)$$

$$\cos((N+1)X) = \cos(NX)\cos(X) - \sin(NX)\sin(X)$$

其中  $X$  为弧度。

这种计算是研究高次谐波的周期现象所必须的。

## 二、程序说明：

程序使用数组  $S(N)$  及工作单元  $X, C, F, U, M, I, N, A, B$ 。

## 三、操作说明：

1. 由BASIC解释程序将本程序输入机内。

2. 键盘命令RUN↵，启动本程序执行，电传印出：

"INPUT X (TO END PROGRAM INPUT 99999), N"

3. 键盘输入  $X$  和最高谐波次数  $N$  值后，电传印出：

"INPUT A, B" 询问一次谐波的系数值  $A_1, B_1$ 。

4. 键盘输入  $A, B$  值，电传又连续印出：

"INPUT A, B" 询问二，三，…， $N$  等各次谐波的系数值

5. 键盘输入  $N-1$  组三角多项式系数值后，电传印出计算结果，并又询问下一次需计算的  $X$  值。

6. 如要继续计算，则重复第3条开始，如不再计算，由键盘输入99999程序结束。

## 五、试题

$$F(X) = \sin(X) + \cos(X) + 2\sin(2X) - 5\cos(2X) + 2\sin(3X) + 3\cos(3X) + \sin(4X) + 5\cos(4X)$$

$$X = 1.25$$

$$F(1.25) = 3.3215$$

## 五、程序清单及运行结果

```

7600 PRINT "INPUT X (TO END PROGRAM INPUT 99999), N"
7602 INPUT X, N
7604 PRINT
7606 DIM S(N)
7608 IF X=99999 THEN GOTO 7652
7610 LET S(1)=SIN(X)
7612 LET C=COS(X)
7614 PRINT "INPUT A, B"
7616 INPUT A, B
7618 PRINT
7620 LET F=A*S(1) + B*C
7622 LET U=S(1)
7624 LET M=C
7626 PRINT "INPUT A, B"

```

```

7628 FOR I=2 TO N
7630 LET S(I)=S(I-1)*M+C*U
7632 LET C=C*M-S(I-1)*U
7634 INPUT A, B
7636 LET F=F+A*S(I)+B*C
7638 NEXT I
7640 PRINT
7642 PRINT "F (", X, ") ="; F
7646 PRINT
7648 PRINT "INPUT X,N"
7650 GOTO 7602
7652 END
运行
RUN
INPUT X (TO END PROGRAM INPUT 99999),N
? 1.25? 4
INPUT A, B
? 1? 1
INPUT A, B
? 2? -5? 2? 3? 1? 5
F (1.25) =3.3215
INPUT X
? 99999
END AT 7652

```

## §8 平面三角形的解法

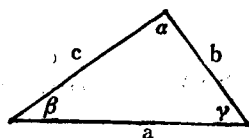
### 一、方法概要:

本程序用以计算三角形的未知边和角, 三角形已给出一边和任何其它两个参数。数据写入的几种可能:

类别	说明
1	角、角、边(AAS)
2	角、边、角(ASA)
3	边、边、角(SSA)
4	边、角、边(SAS)
5	边、边、边(SSS)
所用公式	

$$\text{正弦定理: } \frac{a}{\sin \alpha} = \frac{b}{\sin \beta} = \frac{c}{\sin \gamma}$$

$$\text{余弦定理: } a^2 = b^2 + c^2 - 2bc \cos \alpha$$



(逆时针)  
图 2-8-1

$$b^2 = c^2 + a^2 - 2ca \cos \beta$$

$$c^2 = a^2 + b^2 - 2ab \cos \gamma$$

$$\cos \alpha = \frac{b^2 + c^2 - a^2}{2bc}$$

$$\cos \beta = \frac{c^2 + a^2 - b^2}{2ca}$$

$$\cos \gamma = \frac{a^2 + b^2 - c^2}{2ab}$$

函数关系式:

$$\tan(x) = \frac{\sin(x)}{\cos(x)} = \frac{\sqrt{1 - \cos^2(x)}}{\cos(x)} = \frac{\sin(x)}{\sqrt{1 - \sin^2(x)}}$$

$\alpha, \beta, \gamma$ 三个角的和等于 $180^\circ$ 或 $\pi$ 弧度。

二、程序说明: 程序使用数组A, S, 使用工作单元Z, X。

A为角度值, S为边长的值

输入输出时角度均以“弧度”为单位。

三、操作说明

1. 由BASIC解释程序将本程序送入机内。
2. 键盘命令RUN↵, 启动本程序执行。电传首先询问问题类别。
3. 由键盘输入问题类别代码(1—5), 电传印出要求输入的参数名称和顺序。
4. 由键盘输入三角形的三个已知参数(角度单位为弧度), 其顺序应与第2项中电传印出的顺序相符合。
5. 电传机自动印出计算结果。如果给定条件不合理, 机器印出无解信息: NO SOLUTION并要求重新回答问题类别。重作。  
问题类别代码3, 给定条件为边, 边, 角。在此条件下, 结果可能有三种情况:  
① 有唯一解。  
② 有共轭解, 此时电传印出结果后, 又印出共轭解: ALTERNATE SOLUTION的结果。  
③ 无解, 电传印出NO SOLUTION。
6. 如继续计算, 重复步骤3, 4, 5。如不再计算, 由键盘输入“0”。程序结束。

四、试题:

已知三角形的二个边及一个角

$S_1 = 6, S_2 = 8.53069, A_1 = .5$  (弧度)

求此三角形的第三边及另二角。

解: 求得

$$S_3 = 11.8765$$

$$A_2 = .75 \text{ (弧度)}$$

$$A_3 = 1.89159 \text{ (弧度)}$$

共轭解为:

$$S_3' = 3.09625$$

$$A_2' = 2.39159 \text{ (弧度)}$$

$$A_3' = .25 \text{ (弧度)}$$

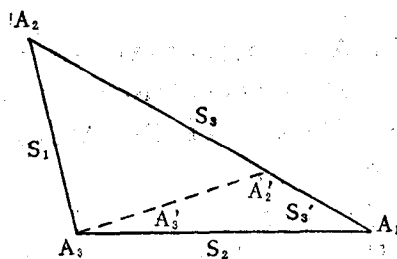


图 2-8-2

## 五、程序清单及运行结果

```
7000 DIM A (3) , S (3)
7002 PRINT "INPUT Z: 0 = END, 1=AAS, 2=ASA, 3=SSA, 4=SAS, 5=
      SSS"
7004 INPUT Z
7006 PRINT
7008 IF Z=0 GOTO 7152
7010 IF Z=1 GOTO 7040
7012 IF Z=2 GOTO 7052
7014 IF Z=3 GOTO 7088
7016 IF Z=4 GOTO 7104
7018 PRINT "INPUT SIDE, SIDE, SIDE"
7020 INPUT S (1), S(2), S (3)
7022 PRINT
7023 IF S(1)+S(2)<=S(3) GOTO 7144
7024 IF S(2)+S(3)<=S(1) GOTO 7144
7026 IF S(3)+S(1)<=S(2) COTO 7144
7028 LET A(1)= (S(2)*S(2)+S(3)*S(3)-S(1)*S(1)) /(2*S(2)*S(3))
7030 LET A(1)=ATN (SQR (1-A(1)*A(1))) /A(1))
7032 IF A(1)>= 0 GOTO 7036
7034 LET A(1)=3.1416+A(1)
7036 GOSUB 7114
7038 GOTO 7002
7040 PRINT "INPUT ANGLE(1), ANGLE(2), SIDE(1)"
7042 INPUT A(1), A(2), S(1)
7044 LET A(3)=3.14159-A(1)-A(2)
7046 LET S(2)=S(1)*SIN (A(2))/SIN (A (1))
7048 LET S(3)=S(1)*SIN (A (3))/SIN(A(1))
7050 GOSUB 7126
7051 GOTO 7002
7052 PRINT"INPUT ANGLE(1), SIDE(3), ANGLE(2)"
7054 INPUT A(1), S(3), A(2)
7056 PRINT
7058 LET A(3)=3.14159-A(1)-A(2)
7060 LET S(1)=S(3)*SIN(A(1))/SIN(A(3))
7062 LET S(2)=S(1)*SIN (A(2))/SIN (A(1))
7064 GOSUB 7126
7066 GOTO 7002
7068 PRINT"INPUT SIDE(1), SIDE(2), ANGLE(1)"
7070 INPUT S(1), S(2), A(1)
```

```

7072 LET A(2)=S(2)*SIN (A(1))/S(1)
7074 IF A(2)>1 GOTO 7144
7076 IF A(2)<1 GOTO 7084
7078 LET S(3)=SQR (S(2)*S(2)-S(1)*S(1))
7080 GOSUB 7114
7082 GOTO 7002
7084 LET X=S(2)*SIN(A(1))
7086 LET Y=SQR(S(1)*(S(1)-X*X)
7088 LET S(3)=SQR(S(2)*S(2)-X*X)+Y
7090 GOSUB 7114
7092 PRINT "ALTERNATE SOLUTION"
7094 LET S(3)=S(3)-2*Y
7096 LET A(3)=A(2)-A(1)
7098 LET A(2)=3.14159-A(2)
7100 GOSUB 7126
7102 GOTO 7002
7104 PRINT "INPUT SIDE(2), ANGLE(1), -SIDE (3)"
7106 INPUT S(2), A(1), S(3)
7108 PRINT
7110 LET S(1)=SQR(S(2)*S(2)+S(3)*S(3)-2*S(2)*S(3)*COS (A(1)))
7111 GOSUB 7114
7112 GOTO 7002
7114 LET A(2)=SIN (A (1)) *S(2)/S(1)
7116 IF A(2)<>1 GOTO 7122
7118 LET A(2)=1.5708
7120 GOTO 7124
7122 LET A(2)=ATN(A(2)/SQR (1-A(2)*A(2)))
7124 LET A(3)=3.14159-A(2)-A(1)
7126 PRINT
7128 FOR I=1 TO 3
7129     IF A(I)<0 GOTO 7146
7130     IF S(I)<0 GOTO 7146
7131 NEXT I
7132 FOR I=1 TO 3
7134     PRINT "SIDE (' , I, ')=' , S(I)
7136     PRINT "OPPOSITE ANGLE (' , I, ')=' , A(I), "RADIANS"
7138 NEXT I
7140 PRINT
7142 RETURN
7144 PRINT

```



```

7146 PRINT 'NO SOLUTION'
7148 PRINT
7150 GOTO 7002
7152 END

```

运行结果:

RUN

INPUT Z: 0 = END, 1 = AAS, 2 = ASA, 3 = SSA, 4 = SAS, 5 = SSS

? 3

INPUT SIDE(1), SIDE(2), ANGLE(1)

? 6? 8.53069? .5

SIDE (1) = 6

OPPOSITE ANGLE(1) = .5 RADIANS

SIDE(2) = 8.53069

OPPOSITE ANGLE(2) = .75 RADIANS

SIDE(3) = 11.865

OPPOSITE ANGLE(3) = 1.89159 RADIANS

ALTERNATE SOLUTION

SIDE(1) = 6

OPPOSITE ANGLE(1) = .5 RADIANS

SIDE(2) = 8.53069

OPPOSITE ANGLE(2) = 2.39159 RADIANS

SIDE(3) = 3.09625

OPPOSITE ANGLE(3) = .25 RADIANS

INPUT Z: 0 = END, 1 = AAS, 2 = ASA, 3 = SSA, 4 = SAS, 5 = SSS

? 5

INPUT SIDE, SIDE, SIDE

? 2? 3? 5

NO SOLUTION

INPUT Z: 0 = END, 1 = AAS, 2 = ASA, 3 = SSA, 4 = SAS, 5 = SSS

? 0

END AT 7152

## §9 求平面多边形面积

### 一、方法概要:

本程序用以计算当给定顶点坐标时, 由相邻两点连线所围成的任意凸多边形的面积。

$$\text{面积} = \frac{1}{2} \{ (X_1 + X_2)(Y_1 - Y_2) + (X_2 + X_3)(Y_2 - Y_3) + \cdots + (X_N + X_1)(Y_N - Y_1) \}$$

式中:  $(X_1, Y_1), (X_2, Y_2), \dots, (X_N, Y_N)$  为连续的各顶点的坐标 (顺时针方向移动)

N——顶点数目。

## 二、程序说明:

程序使用工作单元S, N, X<sub>0</sub>, Y<sub>0</sub>, X, Y, I, A。算出结果后, 工作单元原始数据破坏, 结果直接打印出来, 并保留在A单元。

## 三、操作说明:

1. 由BASIC解释程序将本程序输入机内。
2. 键盘命令RUN启动本程序执行。电传印出:  
"N, X<sub>0</sub>, Y<sub>0</sub>"询问点数和初始顶点坐标。
3. 键盘输入顶点个数N, 初始顶点(X<sub>0</sub>, Y<sub>0</sub>) 电传机再印出"X<sub>1</sub>, Y<sub>1</sub>"询问其它各点坐标值。
4. 键盘输入各顶点的X, Y坐标。读完第N个, 电传机打出印计算结果, 并问是否继续计算, 印出  
"MORE INPUT (1=YES, 0=NO)"
5. 如继续作, 则键盘输入"1", 重复3, 4。如不再计算, 输入"0", 程序结束。
6. 若给定的条件不合理, 求出面积为负值, 则机器印出无解信息"NO SOLUTION"并重新开始执行程序。

## 四、试题:

$$N = 6$$

X <sub>0</sub> = 4	X   4(X <sub>0</sub> )	6	11	12	9	5
Y <sub>0</sub> = 8	Y   8(Y <sub>0</sub> )	11	10	6	4	4

$$\text{结果 } A = 42.5$$

## 五、程序清单及运行结果

```
7700 LET S=0
7702 PRINT "N,X0, Y0"
7704 INPUT N, X0, Y0
7706 PRINT
7708 LET M=N-1
7710 DIM X(M), Y(M)
7712 LET X=X0
7714 LET Y=Y0
7716 PRINT "X1, Y1"
7718 FOR I=1 TO M
7720 INPUT X(I), Y(I)
7722 LET S=S+(X+X(I))*(Y-Y(I))/2
7724 LET X=X(I)
7726 LET Y=Y(I)
7728 NEXT I
7730 PRINT
7732 LET A=(S+(X+X0)*(Y-Y0))/2
7734 IF A>0 THEN GOTO 7740
7736 PRINT "A<0 NO SOLUTION"
```

```

7738 GOTO 7700
7740 PRINT 'A='; A
7742 PRINT 'MORE INPUT (1=YES, 0=NO)'
7744 INPUT K
7746 PRINT
7748 IF K=1 THEN GOTO 7700
7750 END

```

运行结果:

RUN

N, X0, Y0

? 6 ? 4 ? 8

XI                      YI

? 6 ? 11 ? 11 ? 10 ? 12 ? 6 ? 9 ? 4 ? 5 ? 4

A=42.5

MORE INPUT (1=YES, 0=NO)

? 0

END AT 7750

## 第三章 基本复数运算

### §1 求复数自变量的三角函数值

#### 一、方法概要

本程序用以计算 $\sin(Z)$ ,  $\cos(Z)$ ,  $\tan(Z)$ ,  $\sinh(Z)$ ,  $\cosh(Z)$ ,  $\tanh(Z)$ 的值,  
其中 $Z=A+Bi$  ( $A, B$ 为弧度)。

使用公式:

$$\sin(a+bi) = \sin(a)\cosh(b) + i\cos(a)\sinh(b)$$

$$\cos(a+bi) = \cos(a)\cosh(b) + i\sin(a)\sinh(b)$$

$$\tan(a+bi) = \frac{\sin(2a) + i\sinh(2b)}{\cos(2a) + \cosh(2b)}$$

$$\sinh(a+bi) = \sinh(a)\cos(b) + \cos(a)\sin(b)$$

$$\cosh(a+bi) = \cosh(a)\cos(b) + i\sinh(a)\sin(b)$$

$$\tanh(a+bi) = \frac{\sinh(2a) + i\sin(2b)}{\cosh(2a) + \cos(2b)}$$

#### 二、程序说明:

1. 程序使用工作单元 A, B, X, Y, D
2. 初始信息在 A, B 中, 结果信息在 X, Y 中, 印出结果后, X, Y 内容不需要保留。

#### 三、操作说明:

1. 由BASIC解释程序将本程序输入机内
2. 键盘命令RUN启动本程序执行。电传机询问:  
"INPUT A, B (TO END PROGRAM INPUT 0, 0)"
3. 由键盘输入 A, B 的值, 电传机立即打印出结果。
4. 键盘输入 0, 0 程序结束。

#### 四、试题 $Z=2+3i=a+bi$

$$\sin(a+bi)=9.15449-4.16889i$$

$$\cos(a+bi)=4.18961-9.10922i$$

$$\tan(a+bi)=-3.76403E-3+1.00324i$$

$$\sinh(a+bi)=-3.59056+.530926i$$

$$\cosh(a+bi)=3.72455+.511827i$$

$$\tanh(a+bi)=.965386-9.88444E-3i$$

#### 五、程序清单及运行结果

```
7500 PRINT "INPUT A, B (TO END PROGRAM INPUT 0, 0)"
7502 INPUT A, B
7504 PRINT
```

```

7506 IF A * A + B * B = 0 THEN GOTO 7596
7508 DEF FNS(W) = (EXP(W) - EXP(-W)) / 2
7510 DEF FNC(W) = (EXP(W) + EXP(-W)) / 2
7512 LET X = SIN(A) * FNC(B)
7524 LET Y = COS(A) * FNS(B)
7516 IF Y < 0 THEN GOTO 7522
7518 PRINT "SIN(A+BI)=", X, "+", Y, "I"
7520 GOTO 7524
7522 PRINT "SIN(A+BI)=", X, "-", ABS(Y), "I"
7524 LET X = COS(A) * FNC(B)
7526 LET Y = -SIN(A) * FNS(B)
7528 IF Y < 0 THEN GOTO 7534
7530 PRINT "COS(A+BI)=", X, "+", Y, "I"
7532 GOTO 7536
7534 PRINT "COS(A+BI)=", X, "-", ABS(Y), "I"
7536 LET D = COS(2 * A) + FNC(2 * B)
7538 LET X = SIN(2 * A) / D
7540 LET Y = FNS(2 * B) / D
7542 IF Y < 0 THEN GOTO 7548
7544 PRINT "TAN(A+BI)=", X, "+", Y, "I"
7546 GOTO 7550
7548 PRINT "TAN(A+BI)=", X, "-", ABS(Y), "I"
7550 LET X = FNS(A) * COS(B)
7552 LET Y = FNC(A) * SIN(B)
7554 IF Y < 0 THEN GOTO 7560
7556 PRINT "SINH(A+BI)=", X, "+", Y, "I"
7558 GOTO 7562
7560 PRINT "SINH(A+BI)=", X, "-", ABS(Y), "I"
7562 LET X = FNC(A) * COS(B)
7564 LET Y = FNS(A) * SIN(B)
7566 IF Y < 0 THEN GOTO 7572
7568 PRINT "COSH(A+BI)=", X, "+", Y, "I"
7570 GOTO 7574
7572 PRINT "COSH(A+BI)=", X, "-", ABS(Y), "I"
7574 LET D = FNC(2 * A) + COS(2 * B)
7576 LET X = FNS(2 * A) / D
7578 LET Y = SIN(2 * B) / D
7580 IF Y < 0 THEN GOTO 7586
7582 PRINT "TANH(A+BI)=", X, "+", Y, "I"
7584 GOTO 7588

```

```

7586 PRINT "TANH(A+BI)=",X,"-",ABS(Y),"I"
7588 PRINT
7590 PRINT
7592 PRINT "A, B"
7594 GOTO 7502
7596 END

```

运行结果:

```

RUN
INPUT A, B (TO END PROGRAM INPUT 0, 0)

```

? 2 ? 3

```

SIN (A+BI)=9.1545-4.16891 I
COS(A+BI)=-4.18962-9.10922 I
TAN(A+BI)=-3.76403E-3+1.00324 I
SINH(A+BI)=-3.59056+.530922 I
COSH (A+BI) =-3.72454+.511823 I
TANH (A+BI) =.965386-9.88438E-3 I

```

A, B

? -5 ? 8

```

SIN (A+BI) =1429.25+422.791 I
COS (A+BI) =422.791-1429.25 I
TAN (A+BI) =1.22444E-7+1 I
SINH(A+BI)=10.7966+73.4201 I
COSH (A+BI) =-10.7976-73.4135 I
TANH (A+BI) =-1.00009-2.61435E-5 I

```

A, B

? 0 ? 0

END AT 7596

## §2 复数四则运算

### 一. 算法概要:

本程序用来进行形式为  $A+Bi$  的复数按任意顺序排列的  $+$ 、 $-$ 、 $\times$ 、 $\div$  运算。

$N$  = 计算涉及到的复数的个数。

$M$  — 操作代码

运算功能

1

加法

2

减法

3

乘法

4

除法

其计算公式为:

设两个复数为  $A_1 + B_1 i$  和  $A_2 + B_2 i$

加法:  $(A_1 + A_2) + (B_1 + B_2)i$

减法:  $(A_1 - A_2) + (B_1 - B_2)i$

乘法:  $(A_1 A_2 - B_1 B_2) + (A_1 B_2 + B_1 A_2)i$

除法:  $\left( \frac{A_1 A_2 + B_1 B_2}{A_2^2 + B_2^2} \right) + \left( \frac{A_2 B_1 - A_1 B_2}{A_2^2 + B_2^2} \right) i$

计算时不遵守乘、除运算优先的规则按操作代码及复数输入的先后次序顺序计算。

## 二、程序说明:

1. 本程序使用工作单元 N, X, Y, I, M, V, W, A, P。计算得出结果后,部分工作单元的内容将被破坏。
2. 初始信息在 X, Y, V, W 中, 结果在 X, Y 中。

## 三、操作说明:

1. 由 BASIC 解释程序将本程序输入机内。
2. 键盘命令 RUN 启动本程序执行。电传机印出: "INPUT N, X, Y" 询问第一个复数的实部和虚部。  
N——计算涉及到的复数的个数。  
X—— $A_1$   
Y—— $B_1$  } 第一个复数的值。
3. 键盘输入 N, X, Y 值后电传机又印出: "INPUT M, V, W" 询问后续的操作代码和复数的实部和虚部。  
M——加、减、乘、除操作代码  
V—— $A_2, A_3, \dots, A_N$  复数的实部  
W—— $B_2, B_3, \dots, B_N$  复数的虚部
4. 键盘顺序输入  $M_1, A_2, B_2, M_2, A_3, B_3, \dots, M_{N-1}, A_N, B_N$ 。电传机打印出结果, 并询问是否继续计算。
5. 重新计算。键盘输入 "1", 重复步骤 3, 4, 不算再计, 键盘输入 "0", 程序结束。

## 四、试题

求  $(3 + 4i) - (2 - 8i) + (-1 + i) * (0 + i) + (2 + 0i)$

计算结果为:  $8.5 + 5.5i$  (详见程序清单)

## 五、程序清单及运行结果

```
7600 PRINT "INPUT N, X, Y"
7602 INPUT N, X, Y
7604 PRINT
7606 PRINT "INPUT M, V, W"
7608 FOR I=1 TO N-1
7610 INPUT M, V, W
7612 LET A=X
7614 IF M=2 THEN GOTO 7626
```

```

7616 IF M= 3 THEN GOTO 7632
7618 IF M= 4 THEN GOTO 7638
7620 LET X=X+V
7622 LET Y=Y+W
7624 GOTO 7642
7626 LET X=X-V
7628 LET Y=Y-W
7630 GOTO 7642
7632 LET X=X*V-Y*W
7634 LET Y=A*W+V*Y
7636 GOTO 7642
7638 LET X=(X*V+Y*W)/(V*V+W*W)
7640 LET Y=(V*Y-A*W)/(V*V+W*W)
7642 NEXT I
7643 PRINT
7644 IF Y>0 GOTO 7647
7645 PRINT X, "-"; ABS(Y); "I"
7646 GOTO 7648
7647 PRINT X, "+"; Y; "I"
7648 PRINT
7650 PRINT "MORE INPUT (1=YES, 0=NO)"
7652 INPUT P
7654 PRINT
7656 IF P=1 THEN GOTO 7600
7658 END

```

运行结果:

RUN

INPUT N, X, Y

? 5? 3? 4

INPUT M, V, W

? 2? 2? -8? 4? -1? 1? 3? 0? 1? 1? 2? 0

8.5+5.5I

MORE INPUT (1=YES, 0=NO)

? 0

END AT 7658



### §3 求一个复数的平方根

#### 一、方法概要

令复数  $Z = A + iB$  的平方根为:  $\sqrt{A + iB} = x + iy$ ,  $x \geq 0$

将上式两边平方, 其实部与虚部分别为:

$$\begin{cases} x^2 - y^2 = A & (3-3-1) \\ 2xy = B & (3-3-2) \end{cases}$$

规定取  $x \geq 0$  的根, 令  $R = \sqrt{A^2 + B^2}$

解上述联立方程可得:

$$\begin{cases} x = \sqrt{\frac{A+R}{2}} & (3-3-3) \\ y = \pm \sqrt{\frac{R-A}{2}} & (3-3-4) \end{cases}$$

由 (3-3-2) 式可知, 若  $B > 0$ , 则  $x, y$  同号,  $y$  为正,

若  $B < 0$ , 则  $x, y$  异号,  $y$  为负。

#### 二、程序说明

1. 本程序使用工作单元 A, B, R。
2. A 为复数自变量的实部  
B 为复数自变量的虚部  
R 为工作单元
3. 运算结果直接印出, 没有保存 A, R 内容

#### 三、操作说明

1. 用 BASIC 解释程序引入本程序
2. 键盘命令 RUN 启动本程序执行。机器自动印出: "INPUT COMPLEX VALUE (TO END PROG, INPUT 0, 0)" 等待用户输入复数 Z 的 A, B 值。
3. 由键盘输入 A, B 值, 机器立即印出复数的平方根  $ANS.: \times \times \times \times \times \times I$ 。前者为实部后者为虚部
4. 重复询问新的复数值, 并重复 3, 直到 A, B 值输入为 0, 0 则程序转到结束。

#### 四、试题

1.  $Z = 9 + 9i$        $C = \sqrt{Z} = 3.296 + 1.365i$
2.  $Z = 6 - 8i$        $C = \sqrt{Z} = 2.828 - 1.414i$

#### 五、程序清单及运行结果

程序清单:

```
7700 PRINT "INPUT COMPLEX VALUE (TO END PROG, INPUT 0, 0)"
7702 INPUT A, B
7704 PRINT
7706 IF B = 0 THEN GOTO 7712
7708 GOTO 7724
7712 IF A = 0 THEN GOTO 7730
7714 IF A < 0 THEN GOTO 7720
```

```

7716 PRINT "ANS.:", SQR(A)
7718 GOTO 7700
7720 PRINT "ANS.:", SQR(-A); "I"
7722 GOTO 7700
7724 LET R=SQR (A * A+B * B)
7726 PRINT "ANS.:", SQR((A+R)/ 2), SGN(B) * SQR((R-A)/2); "I"
7728 GOTO 7702
7730 END

```

运行结果:

RUN

INPUT COMPLEX VALUE (TO END PROG. INPUT 0,0)

? 9? 9

ANS. 3.29605 1.36527 I

INPUT COMPLEX VALUE (TO END PROG. INPUT 0,0)

? 6? -8

ANS. 2.82843-1.41421 I

INPUT COMPLEX VALUE (TO END PROG. INPUT 0,0)

? 0? 0

END AT 7730

#### §4 复数行列式求值

##### 一、方法概要

本程序对复数行列式

$$\begin{vmatrix} a_{11}+b_{11}j & a_{12}+b_{12}j & \dots & a_{1n}+b_{1n}i \\ a_{21}+b_{21}j & a_{22}+b_{22}j & \dots & a_{2n}+b_{2n}i \\ \dots & \dots & \dots & \dots \\ a_{n1}+b_{n1}j & a_{n2}+b_{n2}j & \dots & a_{nn}+b_{nn}i \end{vmatrix}$$

求值

本程序采用高斯消去法, 先对复数矩阵作变换, 使之成为上三角形阵, 其主对角线上诸元素之乘积即为行列式之值, 具体步骤:

1. 在列中选择绝对值最大的复数, 同时把此复元换到主元位。

$$S = \text{MAX} \{A^2(K, J) + B^2(K, J)\} \quad J = 1, 2, \dots, n.$$

2. 把主元作为基底, 去除同列的各元素。

$$\begin{aligned} A(I, K) + B(I, K)i &= \frac{A(I, K) + B(I, K)i}{A(K, K) + B(K, K)i} \\ &= \frac{A(I, K)A(K, K) + B(I, K)B(K, K)}{A^2(K, K) + B^2(K, K)} + \frac{A(K, K)B(I, K) - A(I, K)B(K, K)}{A^2(K, K) + B^2(K, K)}i \end{aligned}$$

3. 消去下三角形阵

$$A(K, I) = A(K, I) - A(K, I) * A(I, J) + B(K, I) * B(I, J)$$

$$B(K, I) = B(K, I) - B(K, I) * A(I, J) - A(K, I) * B(I, J)$$

4. 对角线元素相乘。

$$\textcircled{1} \begin{cases} S = A(I, I) * A(J, J) - B(I, I) * B(J, J) \\ S_1 = A(I, I) * B(J, J) + A(J, J) * B(I, I) \end{cases}$$

$$I, J = 1, 2, \dots, n$$

$$\textcircled{2} \begin{cases} A^* = A^* * S - B^* * S_1 \\ B^* = B^* * S + A^* * S_1 \end{cases}$$

每计算一次  $S, S_1$ , 代入②一次, 直至结束。

## 二、程序说明

1.  $A(I, J), B(I, J)$  为  $N \times N$  数组的元素分别存放复行列式元素的实部及虚部,  
 $N$ ——行列式阶次, 本程序中  $N \leq 5$ .  
 $E$ ——控制常数, 当某列主元小于  $E$  时, 认为行列式之值为 0.
2. 本程序使用工作单元,  $A, B$  数组,  $N, E, I, I_1, I_3, D_1, D_2, J, J_2, S, T, S_1, X$ .
3. 初始信息在  $A(I, J)$  和  $B(I, J)$  中, 计算结果在  $D_1, D_2$  中, 印出结果后,  $D_1, D_2$  内容即被破坏。

## 三、操作说明:

1. 用 BASIC 解释程序将本程序送入机内
2. 用键盘输入命令 RUN ↵, 启动本程序执行。电传打印出:  
 "ENTER N, E"
3. 由键盘输入矩阵的阶次  $N$ , 控制常数  $E$ , 电传又打印:  
 "ENTER MATRIX  
 COL. 1"
4. 由键盘按列送入矩阵各元素, 先送  $a_{11}$  后送  $b_{11}$ , 第一列送完, 机器自动印出 COL. 2  
 这时送入第二列, 直至全部元素送完, 注意每送一个元素必须按回车键。送完后机器开始运算, 打印结果

" DETERMINENT: "

打印完后, 机器又印出:

" MORE INPUT ( 1 = YES, 0 = NO) "

5. 用户按需要回答, "1" 重新启动本程, 回答 "0" 程序结束

## 四、试题:

$$\begin{vmatrix} 6-i & 5+4i & 4 \\ 5+5i & 2+3i & 0 \\ -1+i & -5+8i & -1+i \end{vmatrix}$$

结果为

DETERINENT: 99-217 i

## 五、程序清单及运行结果

程序清单:

```
7900 DIM A(5, 5), B(5, 5)
7902 PRINT "ENTER N, E"
7904 INPUT N, E
```

```

7906 PRINT
7908 PRINT "ENTER MATRIX"
7910 FOR J = 1 TO N
7912   PRINT "COL." J
7914   FOR J = 1 TO N
7916     INPUT A(J, I), B(J, I)
7918     PRINT
7920   NEXT J
7922 NEXT I
7924 LET D1=1
7926 LET I1=1
7928 LET D2=0
7930 LET I3=I1
7932 LET S=A(I1, I1)*A(I1, I1)+B(I1, I1)*B(I1, I1)
7934 FOR I=I1 TO N
7936   LET T=A(I, I1)*A(I, I1)+B(I, I1)*B(I, I1)
7938   IF S > T THEN GOTO 7944
7940   LET I3=I
7942   LET S=T
7944 NEXT I
7946 IF SQR(S) <= E THEN GOTO 8044
7948 IF I3=I1 THEN GOTO 7966
7950 FOR J=1 TO N
7952   LET S=-A(I1, J)
7954   LET A(I1, J)=A(I3, J)
7956   LET A(I3, J)=S
7958   LET S1=-B(I1, J)
7960   LET B(I1, J)=B(I3, J)
7962   LET B(I3, J)=S1
7964 NEXT J
7966 LET I3=I1+1
7968 FOR I=I3 TO N
7970   LET S1=A(I1, I1)*A(I1, I1)+B(I1, I1)*B(I1, I1)
7972   LET S=(A(I, I1)*A(I1, I1)+B(I, I1)*B(I1, I1))/S1
7974   LET B(I, I1)=(A(I1, I1)*B(I, I1)-A(I, I1)*B(I1, I1))/S1
7976   LET A(I, I1)=S
7978 NEXT I
7980 LET J2=I1-1
7982 IF J2=0 THEN GOTO 7996
7984 FOR J=I3 TO N

```

```

7986   FOR I=1 TO J2
7988       LET A(I1,J)=A(I1,J)-A(I1,I)*A(I,J)+B(I,I)*B(I1,J)
7990       LET B(I1,J)=B(I1,J)-B(I1,I)*A(I,J)-A(I,I)*B(I1,J)
7992   NEXT I
7994 NEXT J
7996 LET J2=I1
7998 LET I1=I1+1
8000 FOR I=I1 TO N
8002     FOR J=1 TO J2
8004         LET A(I, I1)=A(I, I1)-A(I, J)*A(J, I1)+B(I,I)*B(J, I1)
8006         LET B(I, I1)=B(I, I1)-B(I, J)*A(J, I1)-A(I, J)*B(J, I1)
8008     NEXT J
8010 NEXT I
8012 IF I1><N THEN GOTO 7930
8014 LET I3=1
8016 LET J2=INT(N/2)
8018 IF N=2*J2 THEN GOTO 8026
8020 LET I3=0
8022 LET D1=A(N, N)
8024 LET D2=B(N, N)
8026 FOR I=1 TO J2
8028     LET J=N-I+I3
8030     LET S=A(I, I)*A(J, J)-B(I, I)*B(J, J)
8032     LET S1=A(I, I)*B(J, J)+A(J, J)*B(I, I)
8034     LET T=D1*S-D2*S1
8036     LET D2=D2*S+D1*S1
8038     LET D1=T
8040 NEXT I
8042 GOTO 8050
8044 LET D1=0
8046 LET D2=0
8048 GOTO 8058
8050 PRINT
8052 IF D2=>0 THEN GOTO 8060
8054 PRINT "DETERINENT, ",D1,"-";ABS(D2)," I"
8056 GOTO 8062
8058 PRINT
8060 PRINT "DETERINENT, ",D1,"+",D2," I"
8062 PRINT "MORE INPUT (1=YES, 0=NO)"
8064 INPUT X

```

```
8066 IF X=1 THEN GOTO 7902
```

```
8068 END
```

运行结果:

```
RUN
```

```
ENTER N, E
```

```
? 3? 1
```

```
ENTER MATRIX
```

```
COL.1
```

```
? 6? -1
```

```
? 5? 5
```

```
? -1? 1
```

```
COL.2
```

```
? 5? 4
```

```
? 2? 3
```

```
? -5? -8
```

```
COL.3
```

```
? 4? 0
```

```
? 0? 0
```

```
? -1? 1
```

```
DETERINENT: 99-217 I
```

```
MORE INPUT (1=YES, 0 =NO)
```

```
? 0
```

```
END AT 8068
```

## §5 列主元消去法解复线性方程组

### 一、方法概要

本程序用来求解下列复线性代数方程组:

$$(A+iB)(X+iY)=C+iD$$

其中: A和B分别为  $n \times n$  实方阵,  $(X+iY)$  为未知复解向量,  $C+iD$  为复常数向量。

本程序采用复数运算, 对增广矩阵的每列按模最大选取列主元素, 然后进行高斯消去法。

设  $\{U\} = \{A+iB\}$ ,  $\{V\} = \{C+iD\}$ ,  $\{Z\} = \{X+iY\}$

具体步骤如下:

#### 1. 选主元:

$$U_{kk} = \max_{k \leq i \leq n} (|U_{ik}|)$$

#### 2. 正消过程:

对  $k=1, 2, \dots, n$  有:

$$U_{ki} = U_{ki}/U_{kk}, \quad V_k = U_k/U_{kk} \quad j=k, \dots, n$$

$$U_{ij} = U_{ij} - U_{ik} * U_{kj} \quad i=k+1, \dots, n; \quad j=k, \dots, n$$

$$V_i = V_i - U_{i,k} * V_k \quad i = k+1, \dots, n$$

3. 回代过程:

$$Z_n = U_n$$

$$Z_i = U_i - \sum_{j=i+1}^n U_{i,j} * Z_j \quad i = n-1, \dots, 1,$$

## 二、程序说明

1. 本程序使用工作单元, A, B, X, Y 数组及 N, E, I, J, I<sub>0</sub>, C, C<sub>1</sub>, C<sub>2</sub>, T, M, K, T<sub>1</sub>, T<sub>2</sub>, X 等变量。
2. N 为复方程组的阶数, 本程序中  $N \leq 10$ 。  
数组 A(N, N+1), B(N, N+1) 分别存放复增广矩阵元素的实部和虚部。  
数组 X(N), Y(N) 分别存放解向量的实部和虚部。  
E 为控制常数, 当某列元素之模全部小于 E 时, 表明求解失败, 印出 "NO SOLUTION" 重新输入数据。
3. 初始信息在 A(I, J), B(I, J) 中, 运算结果存放在 X(I), Y(I) 中, 待印出结果后, X(I), Y(I) 内容未破坏。

## 三、操作说明:

1. 用 BASIC 解释程序将本程序送入机内。
2. 用键盘输入命令 RUN ↵, 启动本程序执行。电传打印:  
" ENTER N, E "
3. 由键盘输入矩阵阶数 N, 控制常数 E, 电传又打印  
" ENTER AUGMENTED COEFFICIENT MATRIX "  
" COL, 1 "
4. 由键盘输入复方程组增广矩阵各元素, 按列输入, 先送 a<sub>ii</sub> 后送 b<sub>ii</sub>, 每送一个元素必须按回车, 第一列送完, 自动印出: "COL, 2 "

这时输入第二列, 直至全部元素送完, 然后机器运算, 打印结果, 打印完后, 又印出:

" MORE INPUT (1=YES, 0=NO) "

5. 用户根据需要回答, "1" 重新启动本程序; "0" 结束本程序。

## 四 试题:

设有复线性方程组

$$(A + iB)(Z) = (C + iD)$$

其中 A 和 B 为  $n \times n$  实矩阵, 其元素:

$$a_{ki} = b_{kj} = \begin{cases} K-j+1 & (K > j) \\ j & (K \leq j) \end{cases}$$

$C = [C_k]_{n \times 1}$  和  $D = [d_k]_{n \times 1}$  的元素定义如下:

$$\begin{cases} C_k = 0 \\ d_k = \sum_{j=1}^n 2a_{kj} * j \end{cases} \quad (K=1, 2, \dots, n)$$

此方程有以下解向量:

$$[Z] = (1+i, 2+2i, \dots, n+ni)^T$$

若  $n = 2$ , 方程组如下:

$$\begin{bmatrix} 1+i & 1+i \\ 2+2i & 1+i \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 6i \\ 8i \end{bmatrix}$$

计算结果:

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1+i \\ 2+2i \end{bmatrix}$$

##### 五、程序清单及运行结果

```

7200 DIM A(10,11), B(10,11), X(10,11), Y(10,11)
7202 PRINT "ENTER N, E"
7204 INPUT N, E
7206 PRINT
7208 PRINT "ENTER AUGMENTED COEFFICIENT MATRIX"
7210 FOR I=1 TO N+1
7212   PRINT "COL. ", I
7214   FOR J=1 TO N
7216     INPUT A(J, I), B(J, I)
7218     PRINT
7220   NEXT J
7222 NEXT I
7224 LET M=N+1
7226 FOR K=1 TO N
7228   LET C=0
7230   FOR I=K TO N
7232     LET T=A(I,K)*A(I,K)+B(I,K)*B(I,K)
7234     IF C=>T THEN GOTO 7244
7236     LET I0=I
7238     LET C=T
7240     LET C1=A(I, K)
7242     LET C2=B(I, K)
7244   NEXT I
7246   IF SQR(C)=<E THEN GOTO 7322
7248   IF I0=K THEN GOTO 7266
7250   FOR J=K TO M
7252     LET T=A(K, J)
7254     LET A(K, J)=A(I0, J)
7256     LET A(I0,J)=T
7258     LET T=B(K, J)
7260     LET B(K, J)=B(I0, J)
7262     LET B(I0, J)=T
7264   NEXT J
7266   LET C1=C1/C
7268   LET C2=C2/C

```



```

7270     FOR J=K+1 TO M
7272         LET T1=A(K, J)
7274         LET T2=B(K, J)
7276         LET A(K, J)=T1 * C1+T2 * C2
7278         LET B(K, J)=T2 * C1-T1 * C2
7280     NEXT J
7282     FOR I=K+1 TO N
7284         FOR J=K+1 TO M
7286             LET A(I,J)=A(I,J)-A(I,K) * A(K,J)+B(I,K) * B(K,J)
7288             LET B(I,J)=B(I,J)-A(I,K) * B(K,J)-B(I,K) * A(K,J)
7290         NEXT J
7292     NEXT I
7294 NEXT K
7296 FOR I=N TO 1 STEP -1
7298     LET X(I)=A(I, M)
7300     LET Y(I)=B(I, M)
7302     FOR K=1+1 TO N
7304         LET X(I)=X(I)-A(I,K) * X(K)+B(I,K) * Y(K)
7306         LET Y(I)=Y(I)-A(I,K) * Y(K)-B(I,K) * X(K)
7308     NEXT K
7310 NEXT I
7312 FOR I=1 TO N
7314     PRINT "Z(", I, "): RE", X(I), "IM", Y(I)
7316 NEXT I
7318 PRINT
7320 GOTO 7324
7322 PRINT "NO SOLUTION"
7324 PRINT "MORE INPUT (1=YES,0=NO) "
7326 INPUT X
7328 IF X=1 THEN GOTO 7202
7330 GOTO 7334
7332 PRINT "NO SOLUTION"
7334 END

```

运行结果:

RUN

ENTER N, E

? 2 ? 0

ENTER AUGMENTED COEFFICIENT MATRIX

COL. 1

? 1 ? 1

```

? 2? 2
COL. 2
? 1? 1
? 1? 1
COL. 3
? 0? 6
? 0? 8
Z(1): RE 1 IM 1
Z(2): RE 2 IM 2
MORE INPUT(1=YES, 0=NO)
? 0
END AT 7334

```

## §6 复数矩阵的加减法

### 一、方法概要

本程序用来求复数矩阵的和与差。

设  $(X+iY)=(A+iB) \pm (C+iD)$  为  $N \times M$  的复数矩阵, 则有

$$\begin{cases} x_{ij} = a_{ij} \pm c_{ij} \\ y_{ij} = b_{ij} \pm d_{ij} \end{cases} \quad \begin{cases} i=1, 2, \dots, N \\ j=1, 2, \dots, M \end{cases}$$

### 二、程序说明:

1. 本程序使用工作单元: A, B, C, D, X, Y 数组及 M, N, I, J, X, Y 等。
2. 初始信息在 A, B, C, D 数组中, 结果在 X, Y 数组中, 结果印出后, X(I, J), Y(I, J) 中的内容未破坏。

### 三、操作说明:

1. 用BASIC解释程序将本程序送入机内
2. 用键盘命令RUN↵, 启动本程序执行, 电传打印: "ENTER N, M"
3. 用键盘输入矩阵的行数N, 列数M (本程序中  $N \leq 10$ ,  $M \leq 10$ ) 然后电传又印出:  
"ENTER MATRIX [A+iB]"  
"COL.1"
4. 由键盘按列输入矩阵各元素, 先送  $a_{ij}$  后送  $b_{ij}$ , 每送完一个数值后必须按回车键, 送完一列后, 机器自动印出COL.2 这时可输入第二列, 直至全部元素送完电传又印出  
"ENTER MATRIX [C+iD]"  
"COL.1"
- 同 [A+iB] 一样将 [C+iD] 送入, 输入完毕后电传印出:  
"MODE? (1=ADD, 0=SUB)"
5. 用户根据需要回答, "1↵" 表示做加法, "0↵" 表示做减法, 回答以后机器开始运算, 并按列打印结果, 打印完毕又印出:  
"MORE INPUT (1=YES, 0=NO)"

6. 用户根据需要回答,“1↙”表示再运算一次,“0↙”表示结束本程序。

四. 试题:

$$\begin{bmatrix} 2+i2 & 2+i2 \\ 2+i2 & 2+i2 \end{bmatrix} + \begin{bmatrix} 3+i3 & 3+i3 \\ 3+i3 & 3+i3 \end{bmatrix} = \begin{bmatrix} 5+i5 & 5+i5 \\ 5+i5 & 5+i5 \end{bmatrix}$$
$$\begin{bmatrix} 2+i2 & 2+i2 \\ 2+i2 & 2+i2 \end{bmatrix} - \begin{bmatrix} 3+i3 & 3+i3 \\ 3+i3 & 3+i3 \end{bmatrix} = \begin{bmatrix} -1-i & -1-i \\ -1-i & -1-i \end{bmatrix}$$

五. 程序清单及运行结果

```
7000 DIM A(10,10),B(10,10),C(10,10),D(10,10),X(10,10),Y(10,10)
7002 PRINT "ENTER N,M"
7004 INPUT N,M
7006 PRINT
7008 PRINT "ENTER MATRIX(A+IB)"
7010 FOR I=1 TO M
7012 PRINT "COL.",I
7014 FOR J=1 TO N
7016 INPUT A(J,I),B(J,I)
7018 PRINT
7020 NEXT J
7022 NEXT I
7024 PRINT "ENTER MATRIX(C+ID)"
7026 FOR I=1 TO M
7028 PRINT "COL.",I
7030 FOR J=1 TO N
7032 INPUT C(J,I),D(J,I)
7034 PRINT
7036 NEXT J
7038 NEXT I
7040 PRINT "MODE (1=ADD,0=SUB)"
7042 INPUT X
7044 IF X=0 THEN GOTO 7066
7046 FOR I=1 TO N
7048 PRINT
7050 PRINT "COL.",I
7052 FOR J=1 TO M
7054 LET X(J,I)=A(J,I)+C(J,I)
7056 LET Y(J,I)=B(J,I)+D(J,I)
7058 PRINT "RE", X(J,I), "IM", Y(J,I)
7060 NEXT J
7062 NEXT I
7064 GOTO 7080
```

```

7066 FOR I=1 TO N
7068   FOR J=1 TO M
7070     LET C(J,I)=-C(J,I)
7072     LET D(J,I)=-D(J,I)
7074   NEXT J
7076 NEXT I
7078 GOTO 7046
7080 PRINT "MORE INPUT(1=YES,0=NO)"
7082 INPUT Y
7084 PRINT
7086 IF Y=1 THEN GOTO 7002
7088 END

```

运行及结果

RUN

ENTER N,M

? 2? 2

ENTER MATRIX(A+IB)

COL.1

? 2? 2

? 2? 2

COL.2

? 2? 2

? 2? 2

ENTER MATRIX(C+ID)

COL.1

? 3? 3

? 3? 3

COL.2

? 3? 3

? 3? 3

MODE(1=ADD,0=SUB)

? 1

COL.1

RE 5 IM 5

RE 5 IM 5

COL.2

RE 5 IM 5

RE 5 IM 5

MORE INPUT(1=YES,0=NO)

? 1

```

NETER N,M
? 2? 2
ENTER MATRIX(A+IB)
COL.1
? 2? 2
? 2? 2
COL.2
? 2? 2
? 2? 2
ENTER MATRIX(C+ID)
COL.1
? 3? 3
? 3? 3
COL.2
? 3? 3
? 3? 3
MODE(1=ADD,0=SUB)
? 0
COL.1
RE -1 IM -1
RE -1 IM -1
COL.2
RE -1 IM -1
RE -1 IM -1
MORE INPUT(1=YES,0=NO)
? 0
END AT 7088

```

## §7 复数矩阵的乘法

### 一 方法概要:

本程序用于求两个复数矩阵之乘积

若:  $[X+iY] = [A+iB] * [C+iD]$

其中:  $[A+iB]$  为  $N \times M$  矩阵

$[C+iD]$  为  $M \times S$  矩阵

$[X+iY]$  为  $N \times S$  矩阵

则:  $(x_{ij} + iy_{ij}) = \sum_{k=1}^M (a_{ik} + ib_{ik})(c_{kj} + id_{kj})$

即:  $x_{ij} = \sum_{k=1}^M (a_{ik}c_{kj} - b_{ik}d_{kj})$

$y_{ij} = \sum_{k=1}^M (a_{ik}d_{kj} + b_{ik}c_{kj})$

## 二、程序说明

1. 本程序使用工作单元 A, B, C, D, X, Y 数组及 M, N, S, I, J, K, Y。
2. 初始信息在 A, B, C, D 数组中, 计算结果在 X, Y 数组中, 印出结果后, X, Y 数组的内容不需要保留。

## 三、操作说明:

1. 用BASIC解释程序将本程序送入机内
2. 用键盘输入命令RUN启动本程序执行。电传机打印:  
"ENTER N, M, S"
3. 用键盘输入各矩阵的维数 N, M, S (本程序中  $N \leq 10$ ,  $M \leq 10$ ,  $S \leq 10$ ), 然后电传又印出:  
"ENTER MATRIX [A+iB] N×M"  
"COL.1"
4. 由键盘按列输入矩阵各元素, 先送  $a_{ij}$  后送  $b_{ij}$  每送一个数值必须按回车键, 一列输入完毕。机器自动打印出:  
"COL.2"  
这时可输入第二列, 如此进行, 直到全部元素送完, 电传机又打印出:  
"ENTER MATRIX [C+iD] M×S"  
"COL.1"
5. 同输入 [A+iB] 一样, 将 [C+iD] 送入, 然后机器开始运算, 并按列打印结果。打印完毕又印出:  
"MORE INPUT (1=YES, 0=NO)"
6. 用户按需要回答, 回答"1"重新启动本程序。回答"0"则结束本程序。

## 四、试题:

$$\begin{bmatrix} 1+i2 & 1+i2 \\ 1+i1 & 2+i1 \end{bmatrix} * \begin{bmatrix} 1+i1 & 1+i3 \\ 1+i2 & 2+i3 \end{bmatrix} = \begin{bmatrix} -4+i7 & -9+i12 \\ 0+i7 & -1+i12 \end{bmatrix}$$

## 五 程序清单及运行结果

### 程序清单

```
7500 DIM A(10,10),B(10,10),C(10,10),D(10,10),X(10,10),Y(10,10)
7502 PRINT "ENTER N,M,S"
7504 INPUT N,M,S
7506 PRINT
7508 PRINT "ENTER MATRIX(A+IB)"; N; " * "; M
7510 FOR I=1 TO M
7512   PRINT "COL."; I
7514   FOR J=1 TO N
7516     INPUT A(J,I),B(J,I)
7518     PRINT
7520   NEXT J
```

```

7522 NEXT I
7524 PRINT "ENTER MATRIX(C+ID)", M, "*", S
7526 FOR I=1 TO S
7528   PRINT "COL.", I
7530   FOR J=1 TO M
7532     INPUT C(J,I),D(J,I)
7534     PRINT
7536   NEXT J
7538 NEXT I
7540 FOR I=1 TO S
7542   PRINT
7544   PRINT "COL.", I
7546   FOR J=1 TO N
7548     LET X(J,I)=0
7550     LET Y(J,I)=0
7552     FOR K=1 TO M
7554       LET X(J,I)=X(J,I)+A(J,K)*C(K,I)-B(J,K)*D(K,I)
7556       LET Y(J,I)=Y(J,I)+A(J,K)*D(K,I)+B(J,K)*C(K,I)
7558     NEXT K
7560     PRINT "RE", X(J,I), "IM", Y(J,I)
7562   NEXT J
7564 NEXT I
7566 PRINT "MORE INPUT(1=YES,0=NO)"
7568 INPUT Y
7570 PRINT
7572 IF Y=1 THEN GOTO 7502
7574 END

```

运行结果

RUN

ENTER N,M,S

? 2? 2? 2

ENTER MATRIX (A+IB) 2\*2

COL.1

? 1? 2

? 1? 1

COL.2

? 1? 2

? 2? 1

ENTER MATRIX (C+ID) 2\*2

COL.1

```

? 1? 1
? 1? 2
COL.2
? 1? 3
? 2? 3
COL.1
RE -4 IM 7
RE 0 IM 7
COL.2
RE -9 IM 12
RE -1 IM 12
MORE INPUT(1=YES,0=NO)
? 0
END AT 7574

```

## §8 求复数矩阵的逆矩阵

### 一、方法概要:

本程序是对N阶非奇异复数矩阵  $[A+iB]$  求逆。将复数矩阵拼上单位阵  $[I]$ ，使之成为:

$$[A+iB \ I]$$

再运用高斯-约旦消去法，使用复数运算，使之变成  $[I \ (A+iB)^{-1}]$

具体步骤如下:

1. 矩阵  $[A+iB]$  从第一列开始，选择模最大的复元素然后将此复元连同本行换到主元位置，即使

$$|a_{kk}+ib_{kk}|^2 = \text{MAX}(a_{jk}^2+b_{jk}^2) \quad j=K, \dots, N$$

2. 以主元为基底，去除同行各元素:

$$a_{ki}^{(1)} = (a_{ki}^{(0)} * a_{kk}^{(0)} + b_{ki}^{(0)} * b_{kk}^{(0)}) / ((a_{kk}^{(0)})^2 + (b_{kk}^{(0)})^2)$$

$$b_{ki}^{(1)} = (b_{ki}^{(0)} * a_{kk}^{(0)} - a_{ki}^{(0)} * b_{kk}^{(0)}) / ((a_{kk}^{(0)})^2 + (b_{kk}^{(0)})^2)$$

3. 消去主元所在列除主元外的整个一列元素。即:

$$a_{ij}^{(2)} + ib_{ij}^{(2)} = a_{ij}^{(1)} + ib_{ij}^{(1)} - (a_{ik}^{(1)} + ib_{ik}^{(1)}) * (a_{ki}^{(1)} + ib_{ki}^{(1)})$$

分开实部与虚部即:

$$a_{ij}^{(2)} = a_{ij}^{(1)} - a_{ik}^{(1)} * a_{ki}^{(1)} + b_{ik}^{(1)} * b_{ki}^{(1)}$$

$$b_{ij}^{(2)} = b_{ij}^{(1)} - a_{ik}^{(1)} * b_{ki}^{(1)} - b_{ik}^{(1)} * a_{ki}^{(1)}$$

这三个步骤重复进行，直到矩阵  $[A+iB \ I]$  变成  $[I, (A+iB)^{-1}]$ 。



## 二、程序说明:

1. 本程序使用工作单元: A B 数组及 N, E, I, J, K, F, H, C, T, 10, C<sub>4</sub>, X 工作单元。
2. N 为复数矩阵之阶数。本程序中  $N \leq 10$ , E 为控制参数, 当某列主元之模小于 E 时, 认为矩阵奇异, 不能求逆。印出: MATRIX SINGULAR 后重新问话。数组 A(N, 2N), B(N, 2N) 分别存放复矩阵  $[A + iB]$  之实部及虚部。

## 三、操作说明:

1. 用 BASIC 解释程序将本程序送入机内
2. 用键盘输入命令 RUN ↵, 启动本程序执行。电传机印出:  
"ENTER N, E"
3. 由键盘输入矩阵阶次 N, 控制参数 E, 电传又打印:  
"ENTER MATRIX"  
"COL. 1"
4. 由键盘按列输入矩阵元素。先送 A, 后送 B。送完一个数值必须按回车。第一列送完, 机器自动打印:  
"COL. 2"  
这时可送入第二列, 直至全部元素输入完毕, 然后机器开始运算, 打印结果, 打印完后, 机器又印出:  
"MORE INPUT (1=YES, 0=NO)"
5. 用户据需要回答, "1 ↵" 重新启动本程序, "0 ↵" 则结束本程序。

## 四、试题:

$$A = \begin{bmatrix} 0.3125 - i0.3125 & -0.25 + i0.25 & 0.0625 - i0.0625 \\ -0.25 + i0.25 & 0.5 - i0.5 & -0.25 + i0.25 \\ 0.0625 - i0.0625 & -0.25 + i0.25 & 0.3125 - i0.3125 \end{bmatrix}$$
$$A^{-1} = \begin{bmatrix} 3+3i & 2+2i & 1+i \\ 2+2i & 3+3i & 2+2i \\ 1+i & 2+2i & 3+3i \end{bmatrix}$$

## 五、程序清单及运行结果

```
7600 DIM A(10,20),B(10,20)
7602 PRINT "ENTER N,E"
7604 INPUT N,E
7606 PRINT
7608 PRINT "ENTER MATRIX"
7610 FOR I=1 TO N
7612 PRINT "COL.", I
7614 FOR J=1 TO N
7616 INPUT A(J,I),B(J,I)
7618 PRINT
7620 NEXT J
7622 NEXT I
```

```

7624 LET F1=N+1
7626 LET H1=N+N
7628 FOR I=1 TO N
7630   FOR J=F1 TO H1
7632     LET A(I,J)=0
7634     LET B(I,J)=0
7636   NEXT J
7638 NEXT I
7640 FOR I=1 TO N
7642   LET J=I+N
7644   LET A(I,J)=1
7646 NEXT I
7648 FOR K=1 TO N
7650   LET C=0
7652   FOR I=K TO N
7654     LET T=A(I,K)*A(I,K)+B(I,K)*B(I,K)
7656     IF C>T THEN GOTO 7662
7658     LET I0=I
7660     LET C=T
7662   NEXT I
7664   IF SQR(C)=<E THEN COTO 7750
7666   IF I0=K THEN GOTO 7682
7668   FOR J=K TO H1
7670     LET T=A(K,J)
7672     LET A(K,J)=A(I0,J)
7674     LET A(I0,J)=T
7676     LET T=B(K,J)
7678     LET B(K,J)=B(I0,J)
7680     LET B(I0,J)=T
7681   NEXT J
7682   FOR J=K+1 TO H1
7684     LET T=(A(K,J)*A(K,K)+B(K,J)*B(K,K))/C
7686     LET B(K,J)=(B(K,J)*A(K,K)-A(K,J)*B(K,K))/C
7688     LET A(K,J)=T
7690   NEXT J
7692   IF K=1 THEN GOTO 7712
7694   LET C4=K-1
7696   FOR I=1 TO C4
7698     FOR J=K+1 TO H1
7700       LET T=A(I,J)-A(I,K)*A(K,J)+B(I,K)*B(K,J)

```

```

7702     LET B(I,J)=B(I,J)-A(I,K)*B(K,J)-B(I,K)*A(K,J)
7704     LET A(I,J)=T
7706     NEXT J
7708     NEXT I
7710     IF K=N THEN GOTO 7726
7712     FOR I=K+1 TO N
7714         FOR J=K+1 TO H1
7716             LET T=A(I,J)-A(I,K)*A(K,J)+B(I,K)*B(K,J)
7718             LET B(I,J)=B(I,J)-A(I,K)*B(K,J)-B(I,K)*A(K,J)
7720             LET A(I,J)=T
7722         NEXT J
7724     NEXT I
7726     NEXT K
7728     FOR J=1 TO N
7730         PRINT
7732         PRINT 'COL',J
7734         FOR I=1 TO N
7736             LET K=J+N
7738             LET A(I,J)=A(I,K)
7740             LET B(I,J)=B(I,K)
7742             PRINT 'RE', A(I,J), 'IM', B(I,J)
7744         NEXT I
7746     NEXT J
7748     GOTO 7752
7750     PRINT 'MATRIX SINGULAR'
7752     PRINT 'MORE INPUT (1=YES,0=NO)'
7754     INPUT X
7756     PRINT
7758     IF X=1 THEN GOTO 7602
7760     END

```

运行结果:

ENTER N,E

? 3? 0

ENTER MATRIX

COL.1

? 3? 3

? 2? 2

? 1? 1

COL.2

? 2? 2

```

? 3? 3
? 2? 2
COL.3
? 1? 1
? 2? 2
? 3? 3
COL.1
RE .3125      IM  -.3125
RE -.25       IM   .25
RE 6.24999E-2 IM -6.24999E-2
COL.2
RE -.25       IM   .25
RE .5         IM  -.5
RE -.25       IM   .25
COL.3
RE 6.24 999E-2 IM -6.24999E-2
RE -.25       IM   .25
RE .3125      IM  -.3125
MORE INPCT (1=YES,0=NO)
? 1
ENTER N,E
? 3? 0
ENTER MATRIX
COL.1
? .3125? -.3125
? -.25? .25
? .0625? -.0625
COL.2
? -.25? .25
? .5? -.5
? -.25? .25
COL.3
? .0625? -.0625
? -.25? .25
? .3125? -.3125
COL.1
RE 3      IM 3
RE 2      IM 2
RE .999999 IM .999999
COL.2

```

```

RE 2      IM 2
RE 3      IM 3
RE 2      IM 2
COL.3
RE .999999 IM 999999
RE 2      IM 2
RE 3      IM 3
MORE INRUT (1=YES,0=NO)
2 0
END AT 7760

```

### 参 考 资 料

1. 〈电子计算机常用算法〉

科学出版社 1976

2. 〈计算方法〉

武汉大学、山东大学、人民教育出版社

3. 〈计算方法〉

清华大学、北京大学、科学出版社

## 第四章 布尔代数运算

### §1 求逻辑与, 或, 异或

#### 一、计算方法概要

二进制码的逻辑运算是按位进行的, 对于“与”运算, 有下列基本关系 (与运算用 “ $\times$ ” 来表示),

$$1 \times 1 = 1$$

$$1 \times 0 = 0$$

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

例如1101001和0101101相与, 应为:

$$\begin{array}{r} 1101001 \\ 0101101 \\ \hline 0101001 \end{array}$$

对于或运算, 有下列基本关系 (或运算用 “ $+$ ” 来表示)

$$1 + 1 = 1$$

$$1 + 0 = 1$$

$$0 + 0 = 0$$

$$0 + 1 = 1$$

例如上述两个二进制数的“或”运算为

$$\begin{array}{r} 1101001 \\ 0101101 \\ \hline 1101101 \end{array}$$

对于“异或”运算, 有下列关系 (异或运算用 “ $\oplus$ ” 表示)。

$$1 \oplus 1 = 0$$

$$1 \oplus 0 = 1$$

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

例如上述两个二进制数的“异或”运算, 应为:

$$\begin{array}{r} 1101001 \\ \oplus 0101101 \\ \hline 1000100 \end{array}$$

#### 二、程序说明

1. 本程序可对任意字长的二进制数求与、或、异或

2. 程序使用变量:

A (N) 存放参加运算的一个二进制数,

B (N) 存放参加运算的另一个二进制数。

C (N) 存放运算结果。

N 二进制数的位数。

3. 本程序使用的工作单元, I, X, S

#### 三、上机操作步骤:

1. 将本程序送入机内, 由键盘命令 RUN 启动本程序执行。机器自动打印出:

"INPUT THE WORD LENTH"

这时用户应回答二进制数的字长(位数)

2. 回答后, 机器又打印出:

"INPUT THE VALUES OF A AND B"

"A=? "

这时用户应将第一个二进制数从高位到低位按位输入, 每输入一位都要按回车键, 如果输入的不是二进制数, 则机器打印出错误信息

"ERROR, INPUT BINARY NUMBER",

这时用户应重新输入这一位的二进制数。按上述方法输入第二个二进制数。又印出

"PLEASE TYPE IN 1 IF AND, 0 IF OR AND - 1 IF XOR"

这时用户根据需要, 回答 1, 要求进行与运算; 回答 0, 要求进行或运算, 回答 - 1, 则要求进行异或运算

3. 回答后, 机器自动运行, 并打出结果。

A AND B = \*

A OR B = \*

A XOR B = \*

同时印出:

"ANY MORE? PLEASE TYPE IN 1 FOR YES OR 0 FOR NO?"

用户若回答 1, 从头开始运算, 回答 0, 程序结束。

#### 四、例题:

1. A = 1101, B = 1011

A × B = 1001

2. A = 1101 B = 1111

A + B = 1111

3. A = 1101 B = 1011

A ⊕ B = 0110

#### 五、程序清单及运行结果

```
6000 PRINT "INPUT THE WORD LENTH";
6002 INPUT N
6004 PRINT
6006 DIM A(N), B(N), C(N)
6008 PRINT "INPUT THE VALUES OF A AND B"
6010 PRINT "A=";
6012 FOR I=N TO 1 STEP - 1
6013   INPUT A(I)
6014   IF A(I) = 0 GOTO 6018
6015   IF A(I) = 1 GOTO 6018
6016   PRINT "ERROR, INPUT BINARY NUMBER"
6017   GOTO 6013
6018 NEXT I
```

```

6019 PRINT
6020 PRINT "B=";
6022 FOR I=N TO 1 STEP - 1
6023   INPUT B(I)
6024   IF B(I) = 0 GOTO 6028
6025   IF B(I) = 1 GOTO 6028
6026   PRINT "ERROR, INPUT BINARY NUMBER"
6027   GOTO 6023
6028 NEXT I
6029 PRINT
6030 PRINT "PLEASE TYPE IN 1 IF AND, 0 IF OR AND -1 IF XOR";
6032 INPUT X
6034 PRINT
6036 IF X=0 THEN GOTO 6054
6038 IF X=-1 THEN GOTO 6072
6040 PRINT "A AND B=";
6042 FOR I=N TO 1 STEP - 1
6044   LET C(I) = A(I)*B(I)
6046   PRINT C(I);
6048 NEXT I
6050 PRINT
6052 GOTO 6088
6054 PRINT "A OR B=";
6056 FOR I=N TO 1 STEP - 1
6058   LET C(I)=0
6060   IF A(I)+B(I)=0 THEN GOTO 6064
6062   LET C(I)=1
6064   PRINT C(I);
6066 NEXT I
6068 PRINT
6070 GOTO 6088
6072 PRINT "A XOR B=";
6074 FOR I=N TO 1 STEP - 1
6076   LET C(I)=0
6078   IF A(I)=B(I) THEN GOTO 6082
6080   LET C(I)=1
6082   PRINT C(I);
6084 NEXT I
6086 PRINT
6088 PRINT "ANY MORE? PLEASE TYPE IN 1 FOR YES OR 0 FOR NO";

```



```

6090 INPUT S
6092 PRINT
6094 IF S=1 THEN GOTO 6000
6096 END

```

运行结果:

RUN

INPUT THE WORD LENGTH? 4

INPUT THE VALUES OF A AND B

A=? 1 ? 1 ? 0 ? 1

B=? 1 ? 0 ? 1 ? 1

PLEASE TYPE IN 1 IF AND, 0 IF OR AND - 1 IF XOR? 1

A AND B=1 0 0 1

ANY MORE? PLEASE TYPE IN 1 FOR YES OR 0 FOR NO? 1

INPUT THE WORD LENGTH? 4

INPUT THE VALUES OF A AND B

A=? 1 ? 1 ? 0 ? 1

B=? 1 ? 1 ? 1 ? 1

PLEASE TYPE IN 1 IF AND, 0 IF OR AND - 1 IF XOR? 0

A OR B=1 1 1 1

ANY MORE? PLEASE TYPE IN 1 FOR YES OR 0 FOR NO? 1

INPUT THE WORD LENGTH? 4

INPUT THE VALUES OF A AND B

A=? 1 ? 1 ? 0 ? 1

B=? 1 ? 0 ? 1 ? 1

PLEASE TYPE IN 1 IF AND, 0 IF OR AND - 1 IF XOR? -1

A XOR B=0 1 1 0

ANY MORE? PLEASE TYPE IN 1 FOR YES OR 0 FOR NO? 0

END AT 6096

## §2 两个二进制数求和

### 一、计算方法概要

二进制数的加法有下列四种情况

$$0 + 0 = 0$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ (有进位)}$$

$$0 + 1 = 1$$

二进制数的减法有下列四种情况

$$0 - 0 = 0$$

$$0 - 1 = 1 \text{ (有借位)}$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

如果考虑带符号数的运算, 则减法也可看作加法。若用补码表示负数, 则可使带符号数的运算化为简单的加法运算。本程序是采用补码运算的。

## 二、程序说明

1. 程序采用补码算法, 输入输出数据采用原码, 符号单独输入。

2. 程序使用变量

N 参加运算的数的字长

A (N+2) 存放一个运算数, 运行结束时, 存放和

B (N+2) 存放另一个运算数

C 运算过程中的进位

3. 本程序所用的简单变量有: I, Y, J,

4. 本程序不考虑小数点问题。

## 三、程序使用:

1. 将源程序送入机内, 键盘命令 RUN 启动本程序执行。电传打印出:

"INPUT THE MAX WORD LENTH" 这时用户应回答运算数的最大字长

2. 回答后, 又印出:

"INPUT SIGNS OF A AND B: 0 FOR PLUS 1 FOR MINUS"

这时用户应将A和B两个数的符号送入, 正数送0, 负数送1, 每个数之后都要回车。

3. 回答完后, 又印出:

"INPUT THE VALUE OF A AND B"

"A=? "

这时用户应将参加运算的第一个数从高位开始到低位按位输入, 每输入一位都要按回车, 若不够N位, 要在高位补0, 输入时必须是二进制数, 若输入的数不是二进制数, 则打印出错误信息:

"ERROR, INPUT BINARY NUMBER"

用户重新用二进制数将这一位输入, 第一个数送入后, 又印出:

"B=? "

这时用户按上述方法, 将第二个数输入。

4. 全部输入后, 机器自动运行, 并印出结果

"A+B=+ \* \* \* \* " 或 "A+B=- \* \* \* \* "

接着又印出:

"ANY MORE? PLEASE TYPE IN 1 FOR YES OR 0 FOR NO?"

若用户还要运算, 则回答1, 程序重新从头开始运行。否则回答0, 则程序运行结束。

## 四、例题:

已知: A=1001 B=1100

则: A+B=10101

## 五、程序清单及运行结果

### 程序清单

```
6100 REM THIS IS A PROGRAM OF SUMMATION OF TWO BINARY NUMB-
      ERS
6102 PRINT "INPUT THE MAX WORD LENTH";
```

```

6104 INPUT N
6106 PRINT
6108 DIM A(N+2), B(N+2)
6110 PRINT "INPUT THE SIGNS OF A AND B, 0 FOR PLUS, 1 FOR MINUS",
6112 INPUT A(N+2), B(N+2)
6114 PRINT
6116 LET A(N+1)=0
6118 LET B(N+1)=0
6120 PRINT "INPUT THE VALUES OF A AND B"
6122 PRINT "A=";
6124 FOR I=N TO 1 STEP -1
6125   INPUT A(I)
6126   IF A(I)=0 GOTO 6130
6127   IF A(I)=1 GOTO 6130
6128   PRINT "ERROR, INPUT BINARY NUMBER"
6129   GOTO 6125
6130 NEXT I
6131 PRINT
6132 PRINT "B=";
6134 FOR I=N TO 1 STEP -1
6135   INPUT B(I)
6136   IF A(I)=0 GOTO 6140
6137   IF B(I)=1 GOTO 6140
6138   PRINT "ERROR, INPUT BINARY NUMBER"
6139   GOTO 6135
6140 NEXT I
6141 PRINT
6142 IF A(N+2)=0 THEN GOTO 6146
6144 GOSUB 6204
6146 IF B(N+2)=0 THEN GOTO 6160
6148 FOR I=1 TO N+2
6150   LET T=A(I)
6152   LET A(I)=B(I)
6154   LET B(I)=T
6156 NEXT I
6158 GOSUB 6204
6160 LET C=0
6162 FOR I=1 TO N+2
6164   LET A(I)=A(I)+B(I)+C
6166   LET C=0

```

```

6168 IF A(I) < 2 THEN GOTO 6174
6170 LET C = 1
6172 LET A(I) = A(I) - 2
6174 NEXT I
6176 IF A(N + 2) = 0 THEN GOTO 6184
6178 GOSUB 6204
6180 PRINT "A+B="; "-";
6182 GOTO 6186
6184 PRINT "A+B="; "+";
6186 FOR I = N + 1 TO 1 STEP - 1
6188 PRINT A(I);
6190 NEXT I
6192 PRINT
6194 PRINT "ANY MORE? PLEASE TYPE IN 1 FOR YES OR 0 FOR NO";
6196 INPUT Y
6198 PRINT
6200 IF Y = 1 THEN GOTO 6102
6202 END
6204 FOR I = 1 TO N + 1
6206 IF A(I) = 1 THEN GOTO 6210
6208 NEXT I
6210 IF I = N + 1 THEN GOTO 6218
6212 FOR J = I + 1 TO N + 1
6214 LET A(J) = 1 - A(J)
6216 NEXT J
6218 RETURN

```

运行结果:

RUN

INPUT THE MAX WORD LENGTH, 4

INPUT THE SIGNS OF A AND B, 0 FOR PLUS, 1 FOR MINUS ? 0 0

INPUT THE VALUES OF A AND B

A = ? 1 ? 0 ? 0 ? 1

B = ? 1 ? 1 ? 0 ? 0

A+B = + 1 0 1 0 1

ANY MORE? PLEASE TYPE IN 1 FOR YES OR 0 FOR NO? 0

END AT 6202

### §3 求两个二进制数的商

#### 一、方法概要:

本程序采用的是恢复余数法, 即: 将被除数与除数进行一次减法, 由余数的符号来判断够减还是不够减。如果够减商上 1, 如果不够减商上 0, 此时由于不够减而减了, 必须将除数加回去恢复成原来的被除数 (或余数), 除数右移 1 次再作减除数的运算。

步骤: 1. 将被除数和除数由高位到低位逐位分别存入 A (R) 和 B (M) 中

2. 从高位开始 ( $I=1$  到  $M$ ), 按位进行  $A(I) = A(I) - B(I)$

当第  $I$  位相减结果小于 0 时, 将第 1 位到  $I-1$  位的所有余数 (均大于或等于 0) 与第  $I$  位的余数 (小于 0) 相加, 其和若为负, 表示不够减, 商上 0, 并将除数的第 1 至第  $I$  位加回去恢复成原来的被除数 (或余数), 除数右移 1 位; 若前  $I$  位的余数之和大于 0, 表示整体够减, 但第  $I$  位不够减需要向高位借位。当  $M$  位均处理完后, 商上 1, 除数右移一位。重复进行步骤 2 直至得到你所需要的商的有效位数。

3. 同号相除为正, 异号相除为负。A(0), B(0) 放置符号位: 正=0, 负=1, 若  $A(0)-B(0) \approx 0$ , 商为负, 反之为正。A 数组的第  $M+N+1$  位至第  $M+2N$  位放置商。

#### 二、程序说明

1. 本程序使用的工作单元为: A(I), B(I), C(I), N1, M1, N, M, R, H, Q, P。

2. 若进行多次运算, 则 N1 应选为其中所需商的最大的有效位数, M1 应选为其中最大的除数位。例如要进行下列三次运算:

①  $+1100 \div (-11111)$ , 你需要商为五位有效数字。

②  $-11110 \div (+111)$ , 你需要商为十位有效数字。

③  $+1101 \div (+10)$ , 你需要商为四位有效数字。

则应选  $N1=10$ ,  $M1=5$ 。

3. A(0)、B(0) 为符号位 (0=正, 1=负)

在输入除数, 被除数时, 第一位要输入符号位。若进行下列运算:

$+1111 \div (-101)$  所需商的有效位数为 10 位。则  $N=10$ ,  $R=4$ ,  $M=3$ 。

输入:  $A = 0 \ 1111$ ,  $B = 1 \ 101$

R、M 分别表示被除数和除数的位数, 而不包括符号位在内。

N 表示所需商的有效数字的位数, 要求  $N \geq R$ 。

要求被除数第一位不为 0, 这样可以保证你所需的商的有效位数。若前  $r$  位均为 0, 则相应的商的有效位数减为  $N-r$ 。

4. 任何数被 0 除均为无穷大, 所以应避免除数为 0。若输入的除数为 0 时, 本程将打印出  $11 \cdots 1$  共  $N$  位, 其结果是无意义的。

5. 本程序只进行整数除法, 小数点定位问题则由用户自行处理。

#### 三、上机操作步骤:

1. 用 BASIC 解释程序输入本程序

2. 用键盘命令 RUN 启动本程序执行后, 机器自动印出:

"INPUT MAX DIMENSION OF N1 AND M1"

3. 用键盘回答 N1, M1, 机器自动印出:

"INPUT THE NUMBER OF SIGNIFICANT DIGIT OF QUOTIENT NEEDED"

"N=? "

4. 用键盘回答N, 机器接着又打印出:

"INPUT THE DIGIT NUMBER OF DIVISOR AND DIVIDEND"

"M=? "

"R=? "

5. 用键盘分别回答M和R, 机器又打印出:

"INPUT DIVIDEND WITH SIGN (1=NEGATIVE, 0=POSITIVE)"

"A=? "

6. 用键盘输入被除数A, 首先输入符号位(1=负, 0=正), 然后从高位开始, 逐位输入, 每输入一位二进制数, 必须按回车键, 若输入的不是二进制数, 则机器打印出错误信息。

"ERROR, INPUT BINARY NUMBER"

用户重新用二进制数将这一位输入。

7. 用同样方法由键盘输入除数B, 最后机器自动印出商的结果:

A/B= \* \* \* \* \*

正负号    n 位有效数字

其中n位有效数字是以第一个不为0的数开始计起, 至最后一位止。

8. 计算结束后, 机器自动印出问话:

"MORE INPUT (1=YES, 0=NO) "

用户需作出回答, 键盘输入为"1↵", 则重新执行本程序; 若键盘回答为"0↵", 则程序结束。

#### 四、例题

1. 已知A=1110, B=0101

则A÷B=10.1100

2. 已知A=1110, B=1110

则A÷B=1.00000

#### 五、程序清单及运行结果

##### 程序清单

```

6250 REM THIS IS A BINARY DIVISION PROGRAMME
6252 PRINT "INPUT MAX DEMENSION OF N1 AND M1"
6254 PRINT "N1=";
6256 INPUT N1
6258 PRINT
6260 PRINT "M1=";
6262 INPUT M1
6264 DIM A(2*N1+M1+1), B(M1+N1+1), C(M1+N1+1)
6266 PRINT
6268 PRINT "INPUT THE NUMBER OF SIGNIFICANT DIGIT OF QUOTIENT NEEDED"
```

```

6270 PRINT "N=";
6272 INPUT N
6274 PRINT
6276 PRINT "INPUT THE DIGIT NUMBER OF DIVISOR AND DIVIDEND"
6278 PRINT "M=";
6280 INPUT M
6282 PRINT
6284 PRINT "R=";
6286 INPUT R
6288 PRINT
6290 FOR I= 0 TO 2 * N1 + M1
6292     LET A(I)= 0
6294     IF I > M1 + N1 + 1 GOTO 6300
6296     LET B(I)= 0
6298     LET C(I)= 0
6300 NEXT I
6302 PRINT "INPUT DIVIDEND WITH SIGN (1=NEGATIVE, 0=POSITIVE)"
6304 PRINT "A=";
6306 FOR I= 0 TO R
6308     INPUT A(I)
6310     IF A(I)= 0 GOTO 6318
6312     IF A(I)=1 GOTO 6318
6314     PRINT "ERROR, INPUT BINARY NUMBER"
6316     GOTO 6308
6318 NEXT I
6320 PRINT
6322 PRINT "INPUT DIVISOR WITH SIGN(1=NEGATIVE, 0=POSITIVE)"
6324 PRINT "B=";
6326 FOR I= 0 TO M
6328     INPUT B(I)
6330     IF B(I)= 0 GOTO 6338
6332     IF B(I)=1 GOTO 6338
6334     PRINT "ERROR, INPUT BINARY NUMBER"
6336     GOTO 6328
6338 NEXT I
6340 PRINT
6342 LET Z=1
6344 LET Q=M
6346 FOR I=1 TO M
6348     IF B(I)=1 GOTO 6352

```

```

6350 NEXT I
6352 IF I=1 GOTO 6360
6354 FOR J=1 TO M
6356     LET B(J)=B(J+I-1)
6358 NEXT J
6360 FOR J=1 TO N
6362     FOR I=Z TO Q
6364         LET A(I)=A(I)-B(I)
6366         IF A(I)>= 0 GOTO 6388
6368         LET H=I
6370         FOR K=1 TO H
6372             LET C(K)=A(K)+C(K-1)
6374         NEXT K
6376         IF C(H)< 0 GOTO 6392
6378         LET A(H)=1
6380         LET A(H-1)=A(H-1)-1
6382         IF A(H-1)>= 0 GOTO 6388
6384         LET H=H-1
6386         GOTO 6378
6388     NEXT I
6390     GOTO 6402
6392     FOR L=Z TO H
6394         LET A(L)=A(L)+B(L)
6396     NEXT L
6398     IF J<>1 GOTO 6402
6400     LET N=N+1
6402     IF J=N GOTO 6410
6404     FOR L=Q TO Z STEP -1
6406         LET B(L+1)=B(L)
6408     NEXT L
6410     IF C(I)< J GOTO 6414
6412     LET A(N+M+J)=1
6414     LET Q=Q+1
6416     LET Z=Z+1
6418 NEXT J
6420 PRINT "OUTPUT QUOTIENT"
6422 PRINT "A/B=";
6424 IF A(0)-B(0)=0 GOTO 6430
6426 PRINT "-";
6428 GOTO 6432

```



```

6430 PRINT "+";
6432 FOR I=N+M+1 TO 2*N+M
6434 PRINT A(I);
6436 NEXT I
6438 PRINT
6440 PRINT "MORE INPUT? (1=YES,0=NO)"
6442 PRINT "P=";
6444 INPUT P
6446 PRINT
6448 IF P=1 GOTO 6268
6450 END

```

运行结果:

```

INPUT MAX DEMENSION OF N1 AND M1
N1=? 10
M1=? 10
INPUT THE NUMBER OF SIGNIFICANT DIGIT OF QUOTIENT NEEDED
N=? 6
INPUT THE DIGIT NUMBER OF DIVISOR AND DIVIDEND
M=? 4
R=? 4
INPUT DIVIDEND WITH SIGN (1=NEGATIVE, 0=POSITIVE)
A=? 0? 1? 1? 1? 0
INPUT DIVISOR WITH SIGN(1=NEGATIVE, 0=POSITIVE)
B=? 0? 0? 1? 0? 1
OUTPUT QUOTIENT
A/B=+ 1 0 1 1 0 0
MORE INPUT? (1=YES, 0=NO)
P=? 1
INPUT THE NUMBER OF SIGNIFICANT DIGIT OF QUOTIENT NEEDED
N=? 6
INPUT THE DIGIT NUMBER OF DIVISOR AND DIVIDEND
M=? 4
R=? 4
INPUT DIVIDEND WITH SIGN (1=NEGATIVE, 0=POSITIVE)
A=? 0? 1? 1? 1? 0
INPUT DIVISOR WITH SIGN(1=NEGATIVE, 0=POSITIVE)
B=? 0? 1? 1? 1? 0
OUTPUT QUOTIENT
A/B=+ 1 0 0 0 0 0
MORE INPUT? (1=YES, 0=NO)

```

P=0

END AT 6450

说明：在运行结果中不能表示小数点的位置，这由使用者自己加上小数点。

## §4 求两个二进制数的积

### 一、计算方法概要

1. 采用部分积右移方法计算

2. 设乘数是无符号数，被乘数是用补码表示的二进制数，则右移规则为保持最高位（即符号位）不变，把各位数向右移一位，也就是算术右移。

例：01011 → 00101, 1

10010 → 11001, 0

因为右移的目的是将原数除2，这样，在进行乘法运算时，只需注意移位时的符号，其余和两个无符号数乘法一样。

3. 设乘数为补码，则把其变换成无符号数。

因为对一个用补码表示的二进制数 $(X)_{\text{补}}$

$$(X)_{\text{补}} = X_0 X_1 \dots X_{n-1} X_n$$

有：

$$\begin{aligned} X &= -2^n \cdot X_0 + 2^{n-1} \cdot X_1 + 2^{n-2} \cdot X_2 + \dots + 2^1 X_{n-1} + 2^0 X_n \\ &= (-2^n \cdot X_0 + 2^{n-1} \cdot X_1 + 2^{n-1} \cdot X_1) + (-2^{n-1} \cdot X_1 + 2^{n-2} \cdot X_2 + 2^{n-2} \cdot X_2) + \dots \\ &\quad + (-2^1 \cdot X_{n-1} + 2^0 \cdot X_n + 2^0 \cdot X_0) + (-2^0 \cdot X_0 + 0) \\ &= \sum_{i=0}^n (X_{i+1} - X_i) \cdot 2^{n-i}; \quad (\text{式中 } X_{n+1} = 0) \end{aligned} \quad (4-4-1)$$

即对用补码表示的一个二进制数，不论它是正的还是负的，可以造一“数”，其第*i*位之值为原来的补码的第*i*+1位减去第*i*位所得之值，以此作乘数，就可以象无符号数那样进行乘法运算了，只是当乘数某位之值为-1时，应该减去被乘数。

在实际程序中，不是造一“数”，而是根据乘数的第*i*+1位和第*i*位的情况进行判断。

### 二、程序说明：

本程序使用的工作单元：

A(N<sub>1</sub>) 被乘数

B(N<sub>1</sub>+N<sub>2</sub>+2) 乘数和部分积。

N<sub>1</sub>, N<sub>2</sub> 分别为被乘数和乘数的二进制数的位数。

I, J, X, C, K都为循环变量或简单变量。

### 三、上机操作步骤

1. 由BASIC解释程序将本程序送入机内

2. 用键盘命令RUN启动本程序。电传打印出：

“HOW MANY BITS ARE A AND B”

3. 询问被乘数和乘数的位数，分别输入，每输入一个数，要按一下回车键。

4. 回答完后，电传又打印出：

“INPUT A AND B BIT BY BIT”

"A=? "

"B=? "

用户根据具体数从高位逐位将二进制数输入机内，输入时，全部用补码形式输入，最高位（左边）为符号位，若参加运算的数是负数，则为“1”，是正数为“0”。每输入一位应按一个回车键。若输入的数不是二进制数，则输出错误信息：“ERROR, INPUT BINARY NUMBER”，用户则重新用二进制数将这一位输入。

5. 输入完毕，计算机打印计算结果。

"THE PRODUCT IS"

印出的二进制数也是补码形式，最高位为符号位，“1”表示负数，“0”表示正数。

四、例题：

已知：A=1111, B=-1111

则：A×B=-111010001

$(A \times B)_{\text{补}} = 1100010111$

=  
符号位

五、程序清单及运行结果

程序清单

```
6430 LET M=0
6432 REM      "BINARY MULTIPLICATION PROGRAM
6434 REM      THE MULTIPLIER A,B ARE COMPLEMENT
6436 PRINT "HOW MANY BITS ARE A AND B"
6438 INPUT N1, N2
6440 PRINT
6442 DIM A(N1), B(N1+N2+2)
6444 PRINT "INPUT A AND B BIT BY BIT"
6446 PRINT "A=";
6448 FOR I=1 TO N1
6450   GOSUB 6512
6452   LET A(I)=X
6454 NEXT I
6456 LET A(0)=A(1)
6458 PRINT
6460 PRINT "B=";
6462 FOR I=N1+2 TO N1+N2+1
6464   GOSUB 6512
6466   LET B(I)=X
6468 NEXT I
6470 PRINT
6472 LET B (N1+N2+2)=0
```

```

6474 FOR I=1 TO N2
6476 LET J=B(N1+N2+1)*2+B(N1+N2+2)
6478 IF J=0 THEN GOTO 6490
6480 IF J=1 THEN GOTO 6488
6482 IF J=3 THEN GOTO 6490
6484 GOSUB 6550
6486 GOTO 6490
6488 GOSUB 6528
6490 GOSUB 6572
6492 NEXT I
6494 PRINT
6496 PRINT "THE PRODUCT IS"
6498 FOR I=2 TO N1+N2+1
6500 PRINT TAB(I);B(I);
6502 NEXT I
6504 PRINT
6506 IF M=0 THEN GOTO 6510
6508 RETURN
6510 END
6512 INPUT X
6514 IF X=1 THEN GOTO 6526
6516 IF X=0 THEN GOTO 6526
6518 PRINT
6520 PRINT "ERROR, INPUT BINARY NUMBER"
6522 PRINT "THE LAST INPUTTED BIT IS TREATED AS ZERO"
6524 LET X=0
6526 RETURN
6528 REM PLUS SUB-PROGRAM
6530 LET C=0
6532 FOR K=N1+1 TO 1 STEP -1
6534 LET B(K)=B(K)+A(K-1)+C
6536 IF B(K)=<1 THEN GOTO 6544
6538 LET B(K)=B(K)-2
6540 LET C=1
6542 GOTO 6546
6544 LET C=0
6546 NEXT K
6548 RETURN
6550 REM MINUS SUB-PROGRAM
6552 LET C=0

```

```

6554 FOR K=N1+1 TO 1 STEP -1
6556 LET B(K)=B(K)-A(K-1)-C
6558 IF B(K)=>0 THEN GOTO 6566
6560 LET B(K)=B(K)+2
6562 LET C=1
6564 GOTO 6568
6566 LET C=0
6568 NEXT K
6570 RETURN
6572 REM      SHIFT RIGHT PROGRAM
6574 FOR K=N1+N2+2 TO 2 STEP -1
6576 , LET B(K)=B(K-1)
6578 NEXT K
6580 RETURN

```

运行结果:

RUN

HOW MANY BITS ARE A AND B

? 5 ? 6

INPUT A AND B BIT BY BIT

A = ? 0 ? 1 ? 1 ? 1 ? 1

B = ? 1 ? 0 ? 0 ? 0 ? 0 ? 1

THE PRODUCT IS

1 1 0 0 0 1 0 1 1 1 1

END AT 6510

## §5 用卡诺图法化简逻辑表达式

### 一、计算方法概要

本方法是用计算机,按卡诺图化简法对逻辑表达式进行化简,本方法根据布尔代数的基本法则:

$$\bar{A} + A = 1$$

(4-5-1)

$$1 + A = 1$$

(4-5-2)

也就是说:两个与项中,其中只要有一个变量是互补的,而其余变量都相等,这个互补的变量就可以消去,另外,在两个与项中,其中有一个变量只在一个与项中有,而其余各变量都相等,则这个变量也可消去,例如:

$$ABC + \bar{A}BC = AC(B + \bar{B}) = AC$$

其中B变量消去,

$$AB + ABC = AB(1 + C) = AB$$

其中C变量消去。

因为基本BASIC语言不能进行字符串运算,我们将字符用数字代替,如有4个变量, A, B, C, D, 每个变量有两种状态,则用8个数字来代替,  $\bar{A}=1$ ,  $\bar{B}=2$ ,  $\bar{C}=3$ ,  $\bar{D}=4$ , A=5, B=6, C=7, D=8。同时,用数字的组合来代替一个与项,然后按法则(4-5-1), (4-5-2)进行化简。

例如：有卡诺图如下（四个变量）

	$\bar{A} \bar{B}$	$\bar{A} B$	$A \bar{B}$	$A B$
$\bar{C} \bar{D}$	1	0	0	1
$\bar{C} D$	0	0	0	0
$C \bar{D}$	0	0	0	0
$C D$	1	0	0	1

图 4-5-1

将图中的“1”项参加运算：

$$\bar{A} \bar{B} \bar{C} \bar{D} + \bar{A} \bar{B} C D + A \bar{B} C \bar{D} + A \bar{B} C D$$

在机内为：1234+5234+1274+5274

每一项作为一行。

消元方法如下：

1. 取出第K行，如为1, 2, 3, 4, 将此行与其他各行进行比较，比较的原则是：如有n个变量，凡是两行中有n-1列是相同的数字，只有一列不同，则将第K行这一列（不同数字的列）置成0，其他列保持不变，并将处理后的第K行保存起来。（K=1, 2, ..., n-1, n为行数）。
2. 如果第K行与第K+i行比较，所有列的数字全部相同，则将第K+i行全部置成零，保存第K行。（i=1, 2, ..., N-K）
3. 若第K行与其它行比较，相同的列数均小于n-1，则不进行处理，直接将第K行保存起来。
4. 对于最后一行的处理是：如果此行与前面第K行比较，符合1、2两条，则这一行无用，不再参加下一次运算，若符合第3条，则保存此行，参加下一次运算。
5. 对所保存内容，重复1~4步过程，直至不能再简化为止。

如上例：最初的四行四列为：

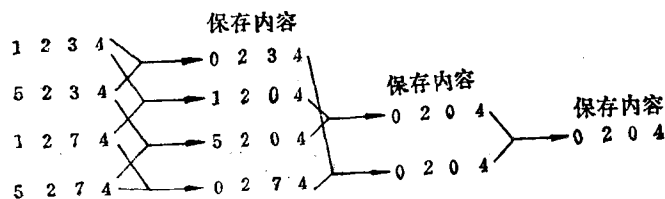


图 4-5-2

最后化简成0204，打印输出时，将数字化成字母输出，凡是某一列的数字为零，不输出，所以输出结果为 $\bar{B} \bar{D}$

本算法只适用于与——或表达式的简化。

## 二、程序说明

本程序使用的工作单元为：

$A(N, M+1)$  为参加比较的两维数组。  
 $B(N, M+1)$  为保存每次比较的中间结果。  
 $L, L_1, L_2, L_3, I, J, V, T$  为工作单元  
 $M$  为变量个数  
 $N$  为最大可能项数。

### 三、上机操作步骤:

1. 将本程序送入机内, 发键盘命令RUN启动本程序执行, 然后, 电传机输出信息, "INPUT THE NUMBER OF VARIABERS"

2. 用户回答卡诺图中的变量数目, 每回答一个数应按“回车”键。回答完后又打印出, "INPUT KARNAUGH MAP IN BINARY NUMBER"

这是为方便用户而设立的。回答时, 以卡诺图中的“1”项作为一个与项, 直接用数字输入, 但只能用二进制数。如:  $\overline{A}\overline{B}\overline{C}\overline{D}$  为0000,  $A\overline{B}C\overline{D}$  为1010。每一个数字要单独作为一个数输入, 有几个变量, 就应输入几个数字, 如1010, 则应先输入左边的数字, 直至全部输入完了为止。第一次输入“1”, 第二次输入“0”, 第三次输入“1”, 第四次输入“0”。如果输入的每一位不是二进制数, 则机器打印出错误信息:

"ERROR, INPUT BINARY NUMBER"

这时用户应用二进制数重新输入这一位。全部输入结束时, 应用“-1”作为结束符输入。

3. 计算机开始计算, 计算完后, 打印结果  
"OUTPUT SIMPLIFIED LOGIC EXPRESSION"

$L = B - D -$

因为 $\overline{B}$ 这个符号没有, 所以我们用 $B-$ 来代替 $\overline{B}$ , 凡是反变量, 全部在字母后面跟一个“-”号, 正变量不变。

### 四、例题

	$\overline{A}\overline{B}$	$\overline{A}B$	$A\overline{B}$	$AB$
$\overline{C}\overline{D}$	1	0	0	1
$\overline{C}D$	0	0	0	0
$C\overline{D}$	0	0	0	0
$CD$	1	0	0	1

图 4-5-3

化简后得:  $L = \overline{B}\overline{D}$

### 五、程序清单及运行结果

#### 程序清单

```

6000 REM TO SIMPLIFY LOGIC EXPRESSION IN
6002 REM KARNAUGH MAP METHOD
6004 PRINT "INPUT THE NUMBER OF VARIABERS";
6006 INPUT M
  
```

```

6008 PRINT
6010 LET N=2↑(M+1)
6012 DIM A(N,M+1), B(N,M+1)
6014 PRINT "INPUT KARNAUGH MAP IN BINARY NUMBER"
6016 LET T=0
6018 FOR I=1 TO N
6020     FOR J=M TO 1 STEP -1
6022         INPUT B(I,J)
6024         IF B(I,J)=-1 GOTO 6040
6026         IF B(I,J)=0 GOTO 6031
6028         IF B(I,J)=1 GOTO 6031
6029         PRINT "ERROR, PLEASE INPUT BINARY NUMBER"
6030         GOTO 6022
6031         LET A(I,J)=J+B(I,J)*M
6032     NEXT J
6033     LET T=T+1
6034     LET A(I,0)=1
6036     PRINT
6038 NEXT I
6040 PRINT
6042 LET L2=T
6044 IF L2=1 THEN GOTO 6158
6046 LET L2=T
6048 LET T=0
6050 LET V=0
6052 FOR K=1 TO L2-1
6054     IF A(K,0)=0 THEN GOTO 6122
6056     LET L3=0
6058     FOR I=K+1 TO L2
6060         IF A(I,0)=0 THEN GOTO 6106
6062         LET L=L+1
6064         FOR J=1 TO M
6066             IF A(K,J)><A(I,J) THEN GOTO 6072
6068             LET L=L+1
6070             GOTO 6074
6072             LET L1=J
6074         NEXT J
6076         IF L<M-1 THEN GOTO 6106
6078         IF L=M-1 THEN GOTO 6090
6080         FOR J=0 TO M

```



```

6082      LET A(I,J)=0
6084      NEXT J
6086      LET A(I,M+1)=1
6088      GOTO 6106
6090      LET L3=L3+1
6092      LET T=T+1
6094      LET A(I,M+1)=1
6096      LET B(T,L1)=0
6098      FOR J=1 TO M
6100          IF J=L1 THEN GOTO 6104
6102          LET B(T,J)=A(K,J)
6104      NEXT J
6106      NEXT I
6108      IF L3><0 THEN GOTO 6122
6110      LET V=V+1
6112      IF A(K,M+1)><0 THEN GOTO 6122
6114      LET T=T+1
6116      FOR J=1 TO M
6118          LET B(T,J)=A(K,J)
6120      NEXT J
6122      NEXT K
6124      IF A(L2,M+1)><0 THEN GOTO 6136
6126      LET V=V+1
6128      LET T=T+1
6130      FOR J=1 TO M
6132          LET B(T,J)=A(L2,J)
6134      NEXT J
6136      FOR I=1 TO T
6138          FOR J=1 TO M
6140              LET A(I,J)=B(I,J)
6142          NEXT J
6144          LET A(I,M+1)=0
6146          LET A(I,0)=1
6148      NEXT I
7150      LET L2=T
6152      IF T=1 THEN GOTO 6156
6154      IF V<T THEN GOTO 6048
6156      PRINT "OUTPUT SIMPLIFIED LOGIC EXPRESSION"
6158      PRINT "L=";
6160      FOR I=1 TO T

```

```

6162   FOR J=1 TO M
6164       IF A(I,J)=0 THEN GOTO 6220
6166       IF A(I,J)=<M THEN GOTO 6172
6168       LET B(I,J)=A(I,J)
6170       LET A(I,J)=A(I,J)-M
6172       IF A(I,J)=1 THEN GOTO 6190
6174       IF A(I,J)=2 THEN GOTO 6194
6176       IF A(I,J)=3 THEN GOTO 6198
6178       IF A(I,J)=4 THEN GOTO 6202
6180       IF A(I,J)=5 THEN GOTO 6206
6182       IF A(I,J)=6 THEN GOTO 6210
6184       IF A(I,J)=7 THEN GOTO 6214
6186       PRINT "H";
6188       GOTO 6216
6190       PRINT "A";
6192       GOTO 6216
6194       PRINT "B";
6196       GOTO 6216
6198       PRINT "C";
6200       GOTO 6216
6202       PRINT "D";
6204       GOTO 6216
6206       PRINT "E";
6208       GOTO 6216
6210       PRINT "F"
6212       GOTO 6216
6214       PRINT "G"
6216       IF B(I,J)>M THEN GOTO 6220
6218       PRINT "-";
6220   NEXT J
6222   IF I>T-1 THEN GOTO 6226
6224   PRINT "+";
6226 NEXT I
6228 PRINT
6230 END

```

程序运行结果:

```

RUN
INPUT THE NUMBER OF VARIABERS ? 4
INPUT KARNAUGH MAP IN BINARY NUMBER
? 0 ? 0 ? 0 ? 0

```

? 0 ? 0 ? 1 ? 0  
 ? 1 ? 0 ? 0 ? 0  
 ? 1 ? 0 ? 1 ? 0  
 ? -1

OUTPUT SIMPLIFIED LOGIC EXPRESSION

L=B-D-

END AT 6230

**本章参考文献:**

(1) 清华大学工业自动化系  
 《数控技术补充教材》

1974.7

(2) 《电子数字计算机》

江苏人民出版社

1973.8

## 第五章 集合运算

### §1 求两个集合的并

#### 一、计算方法概述

定义：令A，B为两个集合，它们的并集合 $A \cup B$ ，是这样一个集合，它的元素至少是属于A和B两者之一。U为“并”运算符。

$A \cup B = \{x \mid x \in A \vee x \in B\}$        $\vee$ 为逻辑“或”运算符      (5—1—1)

根据定义，我们找出所有的属于A和B两者之一的所有元素，组成一个并集合。

#### 二、程序说明

本程序所用的工作单元

A(M) A集合的全部元素

B(N) B集合的全部元素

C(K) 为A，B集合的并集合的全部元素

M，N分别为A，B集合的最多元素个数，X，I，J，K为工作单元

#### 三、上机操作步骤：

1. 用BASIC解释程序将本程序送入机内。

2. 用键盘命令RUN↵，使本程序投入运行，电传打印出：

“INPUT DIMENSIONS M, N”

用户根据要求，输入A集合和B集合的元素个数，M为A集合的元素个数，N为B集合的元素个数。回答一个数要按一下回车键。

3. 回答完后，电传打印出：

“INPUT SET A”

用户输入全部A集合的元素，每输入一个元素要按一下回车键，直至全部输入完毕。

4. 电传又打印出：

“INPUT SET B”

用户输入B集合的全部元素，每输入一个元素要按一下回车键，直至全部输入完毕。

5. 计算机计算后输出：

“OUTPUT THE UNION ON SETS”

紧接着打印出并集合的全部元素。

6. 计算机又打印出：

“MORE INPUT (1=YES, 0=NO)”

若要重复计算，则回答“1”。否则回答“0”，程序结束。

#### 四、试题

已知  $A = \{1, 3, 5, 6\}$

$B = \{2, 3, 4, 5\}$

则： $A \cup B = \{1, 3, 5, 6, 2, 4\}$

## 五、程序清单及运行结果

### 程序清单

```
6000 REM "THE OPREATION OF UNION ON TWO SETS"
6002 PRINT "INPUT DIMENSIONS M,N"
6004 INPUT M,N
6006 PRINT
6007 DIM A(M),B(N),C(M * N)
6008 PRINT "INPUT SET A"
6010 FOR I=1 TO M
6012 INPUT A(I)
6014 NEXT I
6016 PRINT
6018 PRINT "INPUT SET B"
6020 FOR I=1 TO N
6022 INPUT B(I)
6024 NEXT I
6026 PRINT
6028 LET K=0
6030 FOR I=1 TO M
6032 LET C(I)=A(I)
6034 NEXT I
6036 FOR I=1 TO N
6038 FOR J=1 TO M
6040 IF B(I)=A(J) THEN GOTO 6048
6042 NEXT J
6044 LET K=K+1
6046 LET C(M+K)=B(I)
6048 NEXT I
6050 PRINT "OUTPUT THE UNION ON SETS"
6052 FOR I=1 TO M+K
6054 PRINT C(I);
6056 IF I=M+K THEN GOTO 6060
6058 PRINT ", ";
6060 NEXT I
6062 PRINT
6064 PRINT "MORE INPUT(1=YES, 0=NO)"
6066 INPUT X
6068 IF X=1 THEN GOTO 6008
6070 END
```

### 程序运行结果

```

RUN
INPUT DIMENSIONS M,N
? 4 ? 4
INPUT SET A
? 1 ? 3 ? 5 ? 6
INPUT SET B
? 2 ? 3 ? 4 ? 5
OUTPUT THE UNION ON SETS
1, 3, 5, 6, 2, 4
NORE INPUT(1=YES,0=NO)
? 0
END AT 6070

```

## §2 求两个集合的交

### 一、计算方法概要

定义：令  $A$ ， $B$  为两个集合，它们的交集为  $A \cap B$ ，它是这样一个集合，它的元素恰好是既属于  $A$  又属于  $B$  的那些对象。（ $\cap$  为交运算符）

$$A \cap B = \{x \mid x \in A \wedge x \in B\} \quad (5-2-1)$$

根据定义，我们找出所有的既属于  $A$ ，又属于  $B$  的元素，组成一个交集；如果没有这样的元素，则交集为空集合。

### 二、程序说明：

本程序所用的工作单元

$A(M)$ ， $A$  集合的全部元素

$B(N)$ ， $B$  集合的全部元素

$C(K)$ ，为交集的全部元素

$M$ ， $N$  分别为  $A$ ， $B$  集合的最多元素个数。

$I$ ， $J$ ， $K$  为循环变量及计数器。

$X$  为工作单元

### 三、上机操作步骤：

1. 用 BASIC 解释程序将本程序送入机内。

2. 用键盘命令 RUN 使本程序投入运行，电传打印出：

“INPUT DIMENSIONS M,N”

用户根据要求，输入  $A$  集合和  $B$  集合的元素个数， $M$  为  $A$  集合的元素个数， $N$  为  $B$  集合的元素个数，每回答一个数要按一下回车键。

3. 回答完后，电传打印出：

“INPUT SET A”

用户逐个输入  $A$  集合的全部元素，每输入一个元素要按一下回车键，直至全部输入完毕。

4. 电传又打印出：

“INPUT SET B”

用户输入B集合的全部元素,每输入一个元素要按一下回车键,直至全部输入完毕。

5. 计算机开始计算,计算完后打印出:

"OUTPUT INTERSECTION ON SETS"

紧接着打印出交集的全部元素。

6. 最后计算机打印出:

"MORE INPUT (1=YES, 0=NO)"

若要重复计算,则回答"1"。否则回答"0",程序结束。

#### 四 试题

已知:

$A = \{1, 2, 3, 5\}$

$B = \{2, 3, 5, 6\}$

则  $A \cap B = \{2, 3, 5\}$

#### 五、程序清单及运行结果。

程序清单。

```
6080 REM "THE OPREATION OF INTERSECTION ON TWO SETS"
6082 PRINT "INPUT DIMENSIONS M,N"
6084 INPUT M,N
6086 PRINT
6088 DIM A(M),B(N),C(M)
6090 PRINT "INPUT SET A"
6092 FOR I=1 TO M
6094 INPUT A(I)
6096 NEXT I
6098 PRINT
6100 PRINT "INPUT SET B"
6102 FOR I=1 TO N
6104 INPUT B(I)
6106 NEXT I
6108 PRINT
6110 LET K=0
6112 FOR I=1 TO M
6114 FOR J=1 TO N
6116 IF A(I)=B(J) THEN GOTO 6122
6118 NEXT J
6120 GOTO 6126
6122 LET K=K+1
6124 LET C(K)=A(I)
6126 NEXT I
6128 PRINT "OUTPUT THE INTERSECTION ON SETS"
6130 IF K>0 THEN GOTO 6136
```

```

6132 PRINT "C IS NULL-SET"
6134 GOTO 6146
6136 FOR I=1 TO K
6138   PRINT C(I);
6140   IF I=K THEN GOTO 6144
6142   PRINT ", ";
6144 NEXT I
6146 PRINT
6148 PRINT "MORE INPUT(1=YES, 0=NO)"
6150 INPUT X
6152 PRINT
6154 IF X=1 THEN GOTO 6090
6156 END

```

程序运行结果:

```

RUN
INPUT DIMENSIONS M,N
? 4 ? 4
INPUT SET A
? 1 ? 2 ? 3 ? 5
INPUT SET B
? 2 ? 3 ? 5 ? 6
OUTPUT THE INTERSECTION ON SETS
2, 3, 5
MORE INPUT(1=YES,0=NO)
? 0
END AT 6156

```

### §3 求两个集合的相对补

一、计算方法概要:

定义: 令  $A$ 、 $B$  为任意的两个集合,  $A$  相对于  $B$  的补集合是  $A-B$ , 它的元素是把  $A$  中含有的  $B$  的元素全部去掉, 剩下的所有元素, 组成  $A$  相对于  $B$  的补集合。

$$A-B = \{x \mid x \in A \wedge x \notin B\} \quad (5-3-1)$$

根据定义, 我们在  $A$  集合中找出  $B$  集合中也具有的全部元素, 并将它们去掉, 剩下的  $A$  集合中的元素即为相对补集合的全部元素。若  $A$ 、 $B$  集合的元素全部对应, 则相对补集合为空集合。

二、程序说明:

本程序所用的工作单元

$A(M)$   $A$  集合的全部元素

$B(N)$   $B$  集合的全部元素



$C(K)$  为  $A, B$  集合的相对补集合的全部元素。

$M, N$  分别为  $A, B$  集合的最多元素个数。

$I, J, K$  为循环变量及计数器。

$X$  为工作单元

张

### 三、上机操作步骤:

1. 用BASIC解释程序将本程序送入机内。

2. 用键盘命令RUN, 使本程序投入运行。电传打印出:

"INPUT DIMENSIONS M,N"

用户根据要求, 输入A集合和B集合的元素个数,  $M$ 为A集合的元素个数,  $N$ 为B集合的元素个数。回答一个数要按一下回车键。

3. 回答完后, 电传打印出:

"INPUT SET A"

用户输入全部A集合的元素, 每输入一个元素要按一下回车键, 直至全部输入完毕。

4. 电传又打印出:

"INPUT SET B"

用户输入B集合的全部元素, 每输入一个元素要按一下回车键, 直至全部输入完毕。

5. 计算机开始计算, 计算完后输出:

"OUTPUT THE SUBTRACTION ON SETS"

紧接着打印出相对补集合的全部元素。

6. 最后打印出问话:

"MORE INPUT(1=YES, 0=NO)"

若要重复计算, 则回答"1"。否则回答"0", 则程序结束。

### 四、试题:

已知:

$A = \{1, 2, 3\}$

$B = \{2, 3, 4, 5\}$

则  $A - B = \{1\}$

### 五、程序清单及运行结果

程序清单

```
6160 REM "THE OPERATION OF SUBTRACTION ON TWO SETS"
6162 PRINT "INPUT DIMENSIONS M,N"
6164 INPUT M,N
6166 PRINT
6168 DIM A(M),B(N),C(M)
6170 PRINT "INPUT SET A"
6172 FOR I=1 TO M
6174 INPUT A(I)
6176 NEXT I
6178 PRINT
6180 PRINT "INPUT SET B"
```

```

6182 FOR I=1 TO N
6184   INPUT B(I)
6186 NEXT I
6188 PRINT
6190 LET K=0
6192 FOR I=1 TO M
6194   FOR J=1 TO N
6196     IF A(I)=B(J) THEN GOTO 6204
6198   NEXT J
6200   LET K=K+1
6202   LET C(K)=A(I)
6204 NEXT I
6206 PRINT "OUTPUT SUBTRACTION ON SETS"
6208 IF K><0 THEN GOTO 6214
6210 PRINT "C IS NULL-SET"
6212 GOTO 6224
6214 FOR I=1 TO K
6216   PRINT C(I),
6218   IF I=K THEN GOTO 6222
6220   PRINT ", ";
6222 NEXT I
6224 PRINT
6226 PRINT "MORE INPUT(1=YES, 0=NO)"
6228 INPUT X
6230 PRINT
6232 IF X=1 THEN GOTO 6170
6234 END

```

程序运行结果:

```

RUN
INPUT DIMENSIONS M,N
? 3 ? 4
INPUT SET A
? 1 ? 2 ? 3
INPUT SET B
? 2 ? 3 ? 4 ? 5
OUTPUT SUBTRACTION ON SETS
1
MORE INPUT(1=YES,0=NO)
? 0
END AT 6234

```

## §4 求两个集合的笛卡尔积

### 一、计算方法概要:

定义: 任给两个集合  $A, B$ , 我们汇集所有的有序对集合  $\langle x, y \rangle$  (其中  $x \in A, y \in B$ ) 成为一个整体, 即:

$$A \times B = \{Z \mid \exists x \exists y (x \in A \wedge y \in B) \wedge Z = \langle x, y \rangle\} \quad (5-4-1)$$

称为  $A$  和  $B$  的笛卡尔乘积。

根据定义, 我们在  $A$  集合中按顺序逐个取出一个元素, 与  $B$  集合的元素逐个组成有序对, 所有这些有序对的整体, 组成  $A$  和  $B$  的笛卡尔积。

### 二、程序说明:

本程序所用的工作单元

$A(M)$   $A$  集合的全部元素

$B(N)$   $B$  集合的全部元素

$M, N$  分别为  $A, B$  集合的最多元素个数。

$I, J, K$  为循环变量及计数器。

$X$  是工作单元。

### 三、上机操作步骤:

1. 用BASIC解释程序将本程序送入机内。
2. 用键盘命令RUN使本程序投入运行。电传打印出:

"INPUT DIMENSIONS M, N"

用户根据要求, 输入  $A$  集合和  $B$  集合的元素个数, 回答一个数要按一下回车键。

3. 回答完后, 电传打印出:

"INPUT SET A"

用户输入全部的  $A$  集合的元素, 每输入一个元素要按一下回车键, 直至全部输入完毕。

4. 电传又打印出:

"INPUT SET B"

用户输入  $B$  集合的全部元素, 每输入一个元素要按一下回车键, 直至全部输入完毕。

5. 计算机立即输出:

"OUTPUT THE CARTESIAN PRODUCT ON SETS"

紧接着打印出  $A, B$  集合的全部有序对, 这些有序对的全体就是  $A, B$  集合的笛卡尔积。

6. 最后打印出问话:

"MORE INPUT (1=YES, 0=NO)"

若要重复计算, 则回答"1"。否则回答"0", 程序结束。

### 四、试题

已知  $A = \{1, 2\}$

$B = \{3, 4, 5\}$

则 笛卡尔积  $A \times B = \{ \langle 1, 3 \rangle, \langle 1, 4 \rangle, \langle 1, 5 \rangle, \langle 2, 3 \rangle, \langle 2, 4 \rangle, \langle 2, 5 \rangle \}$

## 五、程序清单及运行结果

### 程序清单

```
6242 REM "THE OPREATION OF CARTESIAN PRODUCT ON TWO SETS"
6244 PRINT "INPUT DIMENSIONS M,N"
6246 INPUT M,N
6248 PRINT
6250 DIM A(M),B(N)
6252 PRINT "INPUT SET A"
6254 FOR I=1 TO M
6256 INPUT A(I)
6258 NEXT I
6260 PRINT
6262 PRINT "INPUT SET B"
6264 FOR I=1 TO N
6266 INPUT B(I)
6268 NEXT I
6270 PRINT
6272 PRINT "OUTPUT CARTESIAN PRODUCT ON SETS"
6274 FOR I=1 TO M
6276 FOR J=1 TO N
6278 PRINT "(", A(I), ",", B(J), "),"
6280 NEXT J
6282 NEXT I
6284 PRINT
6286 PRINT "MORE INPUT(1=YES,0=NO)"
6288 INPUT X
6290 PRINT
6292 IF X=1 THEN GOTO 6252
6294 END
```

### 程序运行结果:

```
RUN
INPUT DIMENSIONS M,N
? 2 ? 3
INPUT SET A
? 1 ? 2
INPUT SET B
? 3 ? 4 ? 5
OUTPUT CARTESIAN PRODUCT ON SETS
(1, 3).(1, 4).(1, 5).(2, 3).(2, 4).(2, 5),
MORE INPUT (1 = YES, 0=NO)
```

? 0

END AT 6294

**本章参考文献:**

张锦文《集合论与连续统假设浅说》

上海教育出版社 1979.6.

## 第六章 概率计算

### §1. 计算实验数据的均值、方差并印出经验分布曲线

#### 一、基本概念与算法

在工农业生产, 社会生活和科学研究中, 经常会遇到各种类型的不同实验数据:

$$x_1, x_2, \dots, x_n, \dots, x_N \quad (6-1-1)$$

计算实验数据的数字特征, 就是一种实验数据的处理办法。它为浓缩、简化实验数据提供了经常使用的基本方法。

数字特征是随机变量的一种描述方式, 显然在这里实验数据当作一个随机变量的采样值来处理了。一般我们就取实验数据的算术平均值 $\bar{X}$ 和方差 $s^2$ 两个数字特征。

计算算术平均值的常用算法有:

(1) 直接算法, 即直接使用算术平均值的定义式:

$$\bar{X} = \frac{1}{N} (x_1 + x_2 + \dots + x_N) = \frac{1}{N} \sum_{n=1}^N x_n = \frac{1}{N} \sum_n x_n$$

$$\therefore \text{记为} \quad \bar{X}_{(1)} = \frac{1}{N} \sum_n x_n \quad (6-1-2)$$

(2) 递推算法, 即令 $\bar{X}_0 = 0$

$$\text{则} \bar{X}_n = \frac{n-1}{n} \bar{X}_{n-1} + \frac{1}{n} x_n$$

$$= \bar{X}_{n-1} + \frac{1}{n} (x_n - \bar{X}_{n-1}) \quad n=1, 2, \dots, N$$

$$\therefore \text{记为} \quad \bar{X}_{(2)} = \bar{X}_N \quad (6-1-3)$$

(3) 二次均值算法, 即有

$$\bar{X}_{(3)} = \bar{X}_{(1)} + \frac{1}{N} \sum_{n=1}^N (x_n - \bar{X}_{(1)}) \quad (6-1-4)$$

比较上面三种不同的算法, 不难看出, 直接算法(6-1-2)的运算量最省; 递推算法(6-1-3)可以进行实时处理, 得到一系列的中间均值; 二次均值算法是减少大量实验数据处理的舍入误差, 从而提高精度。

计算方差的常用算法有:

(1) 直接算法, 即

$$s_{(1)}^2 = \frac{1}{N} \sum_{n=1}^N x_n^2 - (\bar{X})^2 \quad (6-1-5)$$

(2) 移去均值算法, 即

$$s_{(2)}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \bar{X})^2 \quad (6-1-6)$$

在我们给出的这段程序中, 计算均值使用递推算法, 计算方差使用直接算法。

实验数据均值 $\bar{X}$ , 方差 $s^2$ 等数字特征, 只是概括地描述了实验数据的一些基本统计特

性。而经验分布曲线可给出实验数据的取值情况。经验分布曲线即为直方图。具体做法如下。

1. 确定实验数据取值上限A和下限B，并决定等分成多少个互不相交的子区间，如取分点： $a_0 < a_1 < a_2 < \dots < a_k < \dots < a_K$

则把  $[a, b]$  分为K个互不相交的小区间  $[a_{k-1}, b_k], k=1, \dots, K$ ，这里  $a_0 \leq a_k, a_k < b$

2. 计算实验数据的经验频率，即检查落入每个子区间上的实验数据的个数  $N_k (k=1, 2, \dots, K)$

$$\text{则有: } f_k = \frac{N_k}{N} \quad (k=1, 2, \dots, K)$$

$$\text{显然 } 0 \leq f_k \leq 1, \sum_{k=1}^K f_k = 1$$

当N充分大时， $f_k$ 可以近似地表示随机变量  $\eta$  在区间  $[a_{k-1}, b_k]$  上取值的概率。由子区间和相应的经验频率组成的曲线叫做经验分布曲线，它给出了随机变量  $\eta$  分布曲线的一个估计值。

## 二、程序说明

程序中从8000~8012，一面输入随机变量的采样值，一面用递推算法取算术平均值。本程序中随机变量为  $[0, 1]$  之间的均匀分布的随机变量。随机变量的采样值放在X(I)数组之中。使用者若要计算别的随机变量的采样值时，改8008句或为键盘输入语句，或为读数据区语句，保证使X(I)数组中放入采样值。8014~8020句计算方差。

从8024~8048，画经验分布曲线图。这段程序除要求原始数据放在X(I)数组中之外，还要数据取值区间的下限A，上限B以及画分布曲线时划分子区间个数S，其它就是计算各个子区间的经验频率值。

本程序使用X(I)一维数组，并使用A、B、C、H、S、S<sub>0</sub>、X<sub>0</sub>等简单变量。

## 三、例题：

本程序中使用的是BASIC语言中给出的  $[0, 1]$  均匀分布的伪随机数

$$\text{即 } f(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{其它} \end{cases}$$

$$\text{则 } F(x) = \begin{cases} 0 & x \leq 0 \\ x & 0 < x \leq 1 \\ 1 & x > 1 \end{cases}$$

$$\therefore E(R) = \int_{-\infty}^{\infty} x f(x) dx = \int_0^1 x dx = \frac{1}{2}$$

$$D(R) = \int_0^1 (x - \frac{1}{2})^2 dx = \frac{1}{12}$$

## 四、上机操作

1. 由BASIC装入本程序，敲RUN/键盘命令启动本程序运行。机器自动询问样本数据个数N之值。
2. 由键盘敲入样本数据个数N之值后，机器自动完成8006~8022的工作，并印出实验数据均值和方差。
3. 然后印出"INPUT A, B AND S"，询问直方图下限值A和上限值B以及子区间

个数 S。由键盘回答以上各值后，机器开始计算经验频率值，并印出经验分布曲线。

#### 五、程序清单和运行结果

##### 程序清单

```
8000 INPUT N
8002 PRINT
8004 DIM X(N)
8006 FOR I=1 TO N
8008   LET X(I)=RND(0)
8010   LET X0=X0+(X(I)-X0)/I
8012 NEXT I
8014 FOR I=1 TO N
8016   LET S0=S0+X(I)*X(I)-X0*X0
8018 NEXT I
8020 LET S0=S0/N
8022 PRINT "MEN.=";X0,"VAR.=";S0
8024 PRINT "INPUT A,B AND S"
8026 INPUT A,B,S
8028 PRINT
8030 LET H=(B-A)/S
8032 FOR K=A TO B STEP H
8034   LET C=0
8036   FOR I=1 TO N
8038     IF X(I)>K THEN GOTO 8044
8040     IF X(I)<K-H THEN GOTO 8044
8042     LET C=C+1
8044   NEXT I
8046   PRINT K;C/N;TAB(15+100*C/N);"+ "
8048 NEXT K
8050 END
```

##### 第一次运行结果

RUN

? 1000

MEN.=.501553 VAR.=8.08619E-2

INPUT A,B AND S

? 0? 1? 5

0	0	+
.2	.195	+
.4	.195	+
.6	.199	+
.8	.218	+



```

1 .193 +
END AT 8050
第二次运行结果
RUN

```

```

? 1000
MEN.=.489947 VAR.=8.28869E-2
INPUT A,B AND S

```

```

? 0? 1? 10
0 0 +
.1 .105 +
.2 .102 +
.3 .093 +
.4 .111 +
.5 .11 + +
.6 .088 +
.7 .107 +
.8 .1 + +
.899999 .089 +
.999999 .095 +
END AT 8050

```

```

第三次运行结果
RUN

```

```

? 2000
MEN.=.502739 VAR.=8.01256E-2
INPUT A,B AND S

```

```

? 0? 1? 10
0 0 +
.1 .0915 +
.2 .0935 +
.3 .0975 +
.4 .105 +
.5 .106 +
.6 .1145 +
.7 .1005 +
.8 .09 +
.899999 .1055 +
.999999 .096 +
END AT 8050

```

## §2 产生任意区间的均匀分布的随机数

### 一、基本概念与算法

基本定理：如果随机变量 $\eta$ 的分布函数 $F(x)$ 连续，则令：

$$R = F(\eta) \quad (6-2-1)$$

则 $R$ 是 $[0, 1]$ 上均匀分布的随机变量。

因为分布函数 $F(x)$ 是在 $[0, 1]$ 上取值的单调递增的连续函数，所以当 $\eta$ 在 $(-\infty, x)$ 内取值时，随机变量 $R$ 则在 $[0, F(x)]$ 内取值，且对应 $[0, 1]$ 上的一个且仅为一个 $r$ 值。所以至少有一个 $x$ 满足：

$$r = F(x) = p\{\eta < x\} \quad (6-2-2)$$

若用 $F_1(r)$ 表示随机变量 $R$ 的分布函数，则有

$$F_1(r) = p\{R < r\} = p\{F(\eta) < r\}$$

$$= \begin{cases} 0 & \text{当 } r \leq 0 \\ p\{\eta < F^{-1}(r)\} = r & \text{当 } 0 < r \leq 1 \\ 1 & \text{当 } r > 1 \end{cases}$$

$\therefore R$ 均匀分布在 $[0, 1]$ 上。用示意图表示以上所述如下(见图6-2-1)：

结论：只要能够产生一个 $[0, 1]$ 之间的均匀分布的随机变量，就可以模拟各种分布的随机变量。在BASIC语言中恰恰设计了一个产生 $[0, 1]$ 之间均匀分布的随机数的标准函数 $RND(x)$ ，经过我们使用实验，其统计特性都比较理想。在本章中所使用的 $[0, 1]$ 之间均匀分布的随机数都是用这个标准函数给出的。

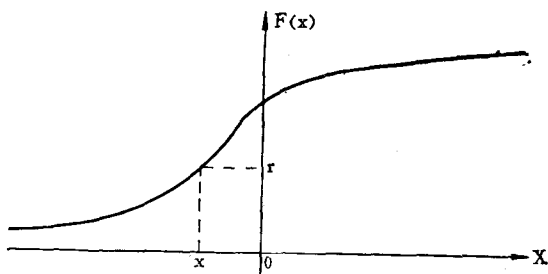


图 6-2-1 随机变量的分布函数图

若要产生 $[a, b]$ 上均匀分布的随机数，计算公式推导如下：

$$\therefore f(x) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & \text{其它} \end{cases}$$

由(6-2-1)式得：

$$R = \int_0^{\eta} \frac{1}{b-a} dx = \frac{\eta - a}{b-a}$$

$$\therefore \eta = (b-a) \cdot R + a \quad (6-2-3)$$

式(6-2-3)就是求 $[a, b]$ 上均匀分布的随机数的计算公式。用BASIC语言就只要一个自定义函数语句就可以了，即

8030 DEF FNA(x) = A + (B-A) \* RND(0)

以下所附的程序就是产生任意区间上均匀分布的随机数的BASIC语言程序，程序中还有求其均值和经验分布曲线，后两者的介绍请见本章第一节所述。

由于程序短小，程序的详细说明及数组，变量说明从略。

二、试题：求  $[3, 5]$  区间上的均匀分布的随机数，显然，其均值为 4。

三、上机操作步骤：

1. 由 BASIC 引入本程序，用 RUN 键盘命令启动本程序执行，机器将自动地印出：“INPUT A,B,N”字样，即询问区间下界 A 值，上界 B 值，以及欲产生随机数的个数 N 之值。

2. 使用者由键盘回答以上各值后，机器印出所求的随机数的均值，印出格式为：

MEN.= \* \* \* \* \*，接着印出经验分布曲线。

说明：由统计学的知识知道，当 N 的值越大，而区间间隔又小，则经验分布曲线越平滑，否则越发凹凸不平。

四、程序清单及运行结果

程序清单

```
8000 PRINT "INPUT A,B,N"
8010 INPUT A,B,N
8020 PRINT
8030 DEF FNA(X)=A+(B-A)*RND(0)
8040 DIM X(N)
8050 LET S=0
8060 FOR I=1 TO N
8070     LET X(I)=FNA(I)
8080     LET S=S+X(I)
8090 NEXT I
8100 LET M=S/N
8110 PRINT TAB (15); "MEN.=", M
8120 FOR I=A TO B STEP 0.2
8130     LET C=0
8140     FOR K=1 TO N
8150         IF X(K)>I GOTO 8180
8160         IF X(K)<I-.2 GOTO 8180
8170         LET C=C+1
8180     NEXT K
8190     PRINT I, TAB (10); C, TAB (15+C/10); "+"
8200 NEXT I
8210 END
```

运行结果：

INPUT A,B,N

? 3? 5? 2000

MEN.=3.99543

3	0	+	
3.2	190		+
3.4	216		+
3.6	187	+	
3.8	210		+
4	192		+
4.2	214		+
4.4	213		+
4.6	202		+
4.8	175	+	
5	201		+

END AT 8210

### §3 产生任意均值和方差的正态分布的随机数

#### 一、基本概念和算法

由概率论的中心极限定理知道,  $N$  个独立的相同分布的随机变量  $R_i$  (其均值为  $\mu_i$ , 方差为  $\sigma_i^2$ ,  $i=1, 2, \dots, N$ ) 之和的概率分布, 当  $N$  很大时, 接近于正态分布, 且其均值和方差为:

$$\begin{aligned}\mu &= \sum_{i=1}^N \mu_i \\ \sigma^2 &= \sum_{i=1}^N \sigma_i^2\end{aligned}\quad (6-3-1)$$

这就是本程序产生任意正态分布的随机数的理论基础。

若我们使用  $[0, 1]$  之间的均匀分布, 则有

$$\mu_i = E(R_i) = \int_0^1 x \cdot 1 dx = \frac{1}{2}, \quad \sigma_i^2 = W_{R_i}(R_i) = \int_0^1 (x - E(x))^2 \cdot 1 dx = \frac{1}{12}$$

$\therefore$  当  $N$  很大时, 就有:

$$\sum_{i=1}^N R_i \sim \mathcal{N}\left(\sum_{i=1}^N \mu_i, \sum_{i=1}^N \sigma_i^2\right) \sim \mathcal{N}\left(\sum_{i=1}^N \frac{1}{2}, \sum_{i=1}^N \frac{1}{12}\right) \sim \mathcal{N}(N/2, N/2) \quad (6-3-2)$$

事实上, 当  $N$  取为 12 时, 近似的程序已是相当好了。本程序就是取  $N=12$  来作的。

由以上所述, 若要产生具有均值为 0, 方差为 1 的标准正态分布的随机变量, 即  $\mathcal{N}(0, 1)$ , 则由概率论的知识知道, 由任意均值和任意方差的正态分布成为标准正态分布, 使用以下公式。

令任意均值和任意方差的正态分布的随机变量为  $Y$ , 其均值为  $\mu$ , 方差为  $\sigma^2$ ; 而令标准正态分布随机变量为  $Z$ , 则有:

$$Z = \frac{Y - \mu}{\sigma} \sim \mathcal{N}(0, 1) \quad (6-3-3)$$

使用 (6-3-2) 的结果, 有:

$$Z = \frac{\sum_{i=1}^N R_i - N/2}{\sqrt{N/12}} \sim \mathcal{N}(0, 1) \quad (6-3-4)$$

若取  $N=12$ , 则 (6-3-4) 式化为:

$$Z = \sum_{i=1}^{12} R_i - 6 \sim \mathcal{N}(0, 1) \quad (6-3-5)$$

式 (6-3-5) 即给出产生标准正态分布随机变量的计算公式。若要产生具有任意均值  $\mu$  和任意方差  $\sigma^2$  的正态分布随机变量  $Y$ , 即  $Y = \mathcal{N}(\mu, \sigma^2)$ , 则使用 (6-3-3) 和 (6-3-4) 式的结果有:

$$\begin{aligned} \frac{Y - \mu}{\sigma} &= \frac{\sum_{i=1}^N R_i - N/2}{\sqrt{N/12}} \\ \therefore Y &= \mu + \sigma \cdot \frac{\sum_{i=1}^N R_i - N/2}{\sqrt{N/12}} \end{aligned} \quad (6-3-6)$$

若取  $N=12$ , 则式 (6-3-6) 化为:

$$Y = \mu + \sigma \cdot \left( \sum_{i=1}^{12} R_i - 6 \right) \quad (6-3-7)$$

式 (6-3-7) 即为本程序产生任意均值和任意方差的正态分布随机变量的计算公式。对比式 (6-3-5) 和式 (6-3-7) 可以看出, (6-3-5) 只是 (6-3-7) 中的一个特例。

## 二、程序说明

本程序使用 BASIC 的标准函数  $RND(X)$  产生  $[0, 1]$  之间的均匀分布随机数。语句 8200~8225 是按公式 (6-3-7) 编出的产生正态分布的随机变量抽样值的子程序。本子程序产生的是均值为 3, 标准差 ( $\sigma$ ) 为 1.5 的正态分布的随机变量抽样值。若要其他任意均值和标准差值, 可改子程序中 8220 句即可。

程序中从 8005 句到 8115 句, 是假想的主程序, 这段程序是控制产生  $N$  个正态分布的随机变量的抽样值, 再求其均值和经验分布曲线并印出, 来验证子程序所求出的随机数的统计特性。

本程序中使用了数组  $X(K)$ , 简单变量  $N, S, C, S_1, Y$  等, 循环变量使用的是  $I, K$ 。

三、试题:  $Y = \mathcal{N}(3, 1.5^2)$

## 四、上机操作步骤:

1. 用 BASIC 解释程序引入本程序, 用  $RUN \swarrow$  键盘命令启动本程序执行, 机器自动印出:

" NPUT NUM. OF DATA N " 询问要产生多少个抽样值。

2. 使用者由键盘敲入  $N$  值之后, 机器自动印出均值, 以及经验分布曲线。

## 五、程序清单及运行结果

### 程序清单

```
8005  PRINT "INPUT NUM OF DATA N"
8010  INPUT N
8015  PRINT
8020  DIM X(N)
8025  LET S=0
8030  FOR K=1 TO N
8035      GOSUB 8200
8040      LET X(K)=Y
8045      LET S=S+X(K)
8050  NEXT K
8055  LET M=S/N
8060  PRINT TAB (15); "MEN.= "; M
8065  PRINT
8070  FOR I=0 TO 6 STEP 0.3
8075      LET C=0
8080      FOR K=1 TO N
8085          IF X(K)>I GOTO 8100
8090          IF X(K)<I-.3 GOTO 8100
8095          LET C=C+1
8100      NEXT K
8105      PRINT I; TAB (10); C; TAB (20+2 * C/10); " + "
8110  NEXT I
8115  END
8200  LET S1=0
8205  FOR I=1 TO 12
8210      LET S1=S1+RND (0)
8215  NEXT I
8220  LET Y=3+1.5 * (S1-6)
8225  RETURN
```

INPUT NUM. OF DATA N

MEN. = 3.01142

**本章参考文献:**

- 1979, 科学出版社

- ## 《Computational Probability and Simulation》

## 第七章 特殊函数求值

### §1 计算贝塞尔函数值

#### 一、方法概要:

本程序用以计算贝塞尔函数  $J_i(x)$  的数值, 其中  $i$  为整数。贝塞尔函数  $J_i(x)$  我们记作:

$$J_i(x) = \left(\frac{x}{2}\right)^i \underbrace{\frac{1}{i!}}_{T_0} \left[ 1 - \underbrace{\frac{1}{1 \cdot (i+1)}}_{T_1} \left(\frac{x}{2}\right)^2 + \underbrace{\frac{1}{2 \cdot (i+2)}}_{T_2} \left(\frac{x}{2}\right)^2 T_1 - \dots \right]$$

其中  $x$  和  $i$  是预先给定的。

#### 二、程序说明:

1. 程序没有使用数组。
2. 程序使用简单变量  $I, X, X_1, S, J, T, L, K$
3. 程序结束原始信息没有破坏, 结果没有寄存。

#### 三、程序使用说明:

1. 由 BASIC 解释程序引入本程序, 键盘命令  $\text{RUN} \checkmark$  启动本程序, 机器自动印出:  
"INPUT I,X,TO END PROG.INPUT 0,0" 要求输入  $I$  和  $X$  值。
2. 由键盘输入  $I$  值和  $X$  值, 机器立即印出结果:  
ANSWER: \* \* \*  
然后程序重复询问新的  $I, X$  值, 直到输入  $0, 0$ , 程序才结束。  
如果输入的  $I$  值不为整数, 则应重新输入  $I$  值 (应为整数)

#### 四、试题、

$$J_1(1) = .440051$$

其中取  $I=1, X=1$ 。

#### 五、程序清单及运行结果

```
√6028 PRINT "INPUT I,X,TO END PROG. INPUT 0,0"  
√6030 INPUT I,X  
√6032 PRINT  
6034 IF I=0 THEN GOTO 6074  
6035 IF I<>INT(I) THEN GOTO 6030  
6036 LET X1=(X/2) * (X/2)  
6038 LET S=1  
6040 LET J=1  
6042 LET T=1  
6044 LET L=1  
6046 LET K=-1
```



```

6048 LET T=T * X1/J/(1+J)
6050 LET S=S+K * T
6052 IF T < .000001 THEN GOTO 6060
6054 LET K=-K
6056 LET J=J+1
6058 GOTO 6048
6060 FOR K=1 TO I
6062 LET L=L * K
6064 NEXT K
6066 PRINT "ANSWER: "; (X/2)↑I/L * S
6068 PRINT
6070 PRINT "INPUT I, X"
6072 GOTO 6030
6074 END

```

六 程序运行结果:

RUN

INPUT I, X, TO END PROG. INPUT 0, 0

? 1? 1

ANSWER: .440051

INPUT I, X

? 0? 0

END AT 6074

## §2 计算 $\gamma$ 函数值

一、方法概要:

令  $T(X) = 1 + b_1 X + b_2 X^2 + \dots + b_8 X^8$

其中  $b_1 = -0.577192$      $b_6 = -0.756702$

$b_2 = 0.988206$      $b_8 = 0.482199$

$b_3 = -0.897057$      $b_7 = -0.193528$

$b_4 = 0.918207$      $b_8 = 0.0358683$

令  $S(P) = X(X+1)(X+2)\dots(X+P-1)$

∴ 当

$X=0$      $\gamma(X)=0$

$X=-1$      $\gamma(X)$ 不存在

$0 \leq X < 1$      $\gamma(X) = \frac{T(X+1)}{X}$

$1 \leq X < 2$      $\gamma(X) = T(X)$

$2 \leq X$  令  $P = \text{INT}(X) - 1$ ,  $\gamma(X) = T(X-P) \cdot S(P)$

$X < 0$  令  $P = \text{ABS}(\text{INT}(X)) + 1$

$$\gamma(X) = \frac{T(X+P+1)}{(X+P)S(P)}$$

## 二、程序说明:

1. 程序未使用数组。
2. 程序使用简单变量 X, P, S, T, Y, I, W, B
3. 程序结束原始输入信息破坏, 结果在 X 变量中。

## 三、程序使用:

1. 由 BASIC 解释程序引入本程序, 键盘命令 RUN 启动本程序执行。机器自动印出,  
"INPUT X. TO END PROG. INPUT 0"

要求输入 X 值。

2. 由键盘输入 X 值, 机器立即印出结果:

GAMMA(X) = \* \* \*

然后继续询问新的 X 值, 直到输入为 0 值, 程序结束。

## 四、试题:

GAMMA (6) = 120

GAMMA(4.5) = 11.6317

GAMMA(1.25) = 0.906402

## 五、程序清单和运行结果

### 程序清单

```

6076 PRINT "INPUT X. TO END PROG. INPUT 0"
6078 INPUT X
6080 PRINT
6082 IF X=0 THEN GOTO 6182
6084 IF INT(X)/ABS(X)><-1 THEN GOTO 6090
6086 PRINT "GAMMA(X) UNGIVED"
6088 GOTO 6174
6090 IF X<2 THEN GOTO 6104
6092 LET P=INT(X)-1
6094 LET X=X-P
6096 GOSUB 6140
6098 GOSUB 6150
6100 LET X=S*T
6102 GOTO 6172
6104 IF X<1 THEN GOTO 6112
6106 GOSUB 6150
6108 LET X=T
6110 GOTO 6172
6112 IF X=0 THEN GOTO 6124
6114 LET Y=X
6116 LET X=X+1

```

```

6118 GOSUB 6150
6120 LET X=T/Y
6122 GOTO 6172
6124 LET P=ABS(INT(X))+1
6126 GOSUB 6140
6128 LET X=X+P
6130 LET Y=X
6132 LET X=X+1
6134 GOSUB 6150
6136 LET X=T/Y/S
6138 GOTO 6172
6140 LET S=1
6142 FOR I=0 TO P-1
6144   LET S=S*(X+I)
6146 NEXT I
6148 RETURN
6150 LET T=1
6152 LET W=1
6154 LET X=X-1
6156 FOR I=1 TO 8
6158   LET W=W*X
6160   READ B
6162   LET T=T+W*B
6164 NEXT I
6166 RETURN
6168 DATA -.577192,.988206,-.897057
6170 DATA .918207,-.756702,.482199,-.193528,3.58683E-2
6172 PRINT "GAMMA(X)=";X
6174 PRINT
6176 PRINT "X",
6178 RESTORE
6180 GOTO 6078
6182 END

```

程序运行结果:

RUN

INPUT X. TO END PROG. INPUT 0

? 6

GAMMA(X)=120

X ? 4.5

GAMMA(X)=11.6317

X ? 1.25

GAMMA(X)=.906403

X ? 0

END AT 6182

### §3 超几何函数求值

#### 一、方法概要:

超几何函数的公式为:

$$M(Z) = \frac{AZ}{B} + \frac{A(A+1)Z^2}{B(B+1)2!} + \dots + \frac{A(A+1)(A+2)\dots(A+N-1)Z^N}{B(B+1)(B+2)\dots(B+N-1)N!}$$

项数N确定结果的精确度, 项数越大, 结果就越精确。

#### 二、程序说明:

1. 程序没使用数组
2. 程序使用简单变量 A, B, Z, N, S, I
3. A, B, Z, N 要求输入, 程序结束, 结果在 S 中, 项数 N 在机器容许数值内, 没有限制。  
程序结束, 原始信息没有破坏。

#### 三、程序使用

1. 由BASIC解释程序将本程序引入机内, 由键盘命令RUN启动本程序执行。机器自动印出:  
“A, B, Z, N? (TO END PROG. INPUT 0, 0, 0, 0)” 询问初始数据或程序要不要结束。
2. 由键盘输入数据 A, B, Z, N 之值后, 机器立即印出结果以及 “A, B, Z, N? 程序进行重复。  
如果要结束程序, 由键盘输入 0, 0, 0, 0 则程序结束。

#### 四、试题:

$$e = 1 + \frac{1}{2} + \frac{1}{3} + \dots$$

$$= 2.71828$$

其 A=1, B=1, Z=1, N=10

#### 五、程序清单及运行结果

程序清单

```
6190 PRINT "A, B, Z, N (TO END PROG. INPUT 0, 0, 0, 0)"
6192 INPUT A, B, Z, N
6194 PRINT
6196 IF N=0 THEN GOTO 6214
6198 LET S=1
6200 FOR I=N TO 1 STEP -1
6202 LET S=1+(A+I-1)*Z*S/((B+I-1)*I)
```

```

6204 NEXT I
6206 PRINT "M=";S
6208 PRINT
6210 PRINT "A, B, Z, N"
6212 GOTO 6192
6214 END

```

程序运行结果:

RUN

A, B, Z, N (TO END PROG. INPUT 0, 0, 0, 0)

? 1? 1? 1? 10

M=2.71828

A, B, Z, N

? 0? 0? 0? 0

END AT 6214

### 本章参考文献

- 〔1〕 M.R.Spiegel 著, 谢国瑞等译  
《高等数学的理论和习题》 上海科技出版社  
1978
- 〔2〕 В.И.СМИРНОВ 《高等数学教程》

## 第八章 高次代数方程求根

### §1 Newton-Raphson 法求多项式的实根

#### 一、方法概要:

把方程  $f(x) = 0$  在  $x = x_i$  附近作台劳展开, 略去二次与二次以上的项得:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) = 0$$

∴ 可以从中得出 Newton-Raphson 迭代公式:

若  $f'(x_i) \neq 0$ ,

$$\text{则 } x_{i+1} = x_i - f(x_i)/f'(x_i)$$

当  $1 - \frac{x_i}{x_{i+1}} < 0.00001$  迭代结束。

选取较好的迭代初值  $x_0$ 。用此公式进行迭代, 可以求出  $f(x)$  的全部实根。

#### 二、程序说明:

1. 程序使用数组 A
2. 程序使用简单变量 N, X, J, I, K, B, G, C, D, L。
3. 程序结束, 原始信息没有破坏, 结果在 X 中。
4. 如果迭代超过 100 次而不收敛, 则程序印出: "AFTER 100 ITERATIONS, NO CONVERGENCE"
5. 本程序不能求重根及复根。

#### 三、程序使用

1. 用 BASIC 解释程序引入本程序, 键盘命令 RUN 启动本程序执行。机器自动印出:  
"INPUT DIGREE OF POLYNOMIAL N" 询问多项式的阶数。
2. 由键盘输入多项式的阶, 机器又自动印出:  
"INPUT COEFFICIONS A(1), ..., A(N+1), (4/LINE)"
3. 这时用户输入各项系数, 缺项时, 其系数用零输入, 输入时从  $x^n$  的高次项系数开始输入, 每行输入 4 个系数。
4. 系数输入完, 机器自动打印出:  
"INPUT YOUR ESTIMATTE OF ROOT"  
这时应输入根的估计值, 作为迭代初值。
5. 然后机器输出结果  
ROOT IS \* \* \*  
然后又印出:  
"ANOTHER ROOT IS? (1=YES, 0=NO)" 机器向用户询问有另外的根吗?  
若有时, 请回答 "1", 程序重复第 4 步输入另一根的估计值, 否则回答 "0", 程序结束。

#### 四、试题:

$$f(x)=x^2-4.01x+4.02=0$$

其根为  $x_1=2$

$x_2=2.01$

## 五、程序清单及运行结果

### 程序清单

```

7860 DIM A (99)
7862 PRINT "INPUT DIGREE OF POLYNOMIAL N"
7864 INPUT N
7866 PRINT
7868 PRINT "INPUT COEFFICIENTS A(1), ..., A(N+1).(4/LINE)"
7870 FOR I=0 TO INT ((N+4)/4)-1
7872   INPUT A(4*I+1), A(4*I+2), A(4*I+3), A(4*I+4)
7874   PRINT
7876 NEXT I
7878 PRINT "INPUT YOUR ESTIMATE OF ROOT"
7880 INPUT X
7882 PRINT
7884 FOR J=1 TO 100
7886   LET F=A(N+1)
7888   FOR I=1 TO N
7890     LET B=1
7892     FOR K=1 TO N-I+1
7894       LET B=B*X
7896     NEXT K
7898     LET F=F+A(I)*B
7900   NEXT I
7902   LET G=A(N)
7904   FOR I=1 TO N-1
7906     LET C=1
7908     FOR D=1 TO N-I
7910       LET C=C*X
7912     NEXT D
7914     LET G=G+(N-I+1)*A(I)*C
7916   NEXT I
7918   LET I=X
7920   LET X=X-F/G
7922   IF ABS(X)+ABS(I)=0 THEN GOTO 7934
7924   IF X=0 THEN GOTO 7928
7926   IF ABS(1-I/X)<.00001 THEN GOTO 7934
7928 NEXT J

```

```

7930 PRINT "AFTER 100 ITERATIONS, NO CONVERGENCE"
7932 GOTO 7936
7934 PRINT "ROOT IS:", X
7936 PRINT "ANOTHER ROOT IS?(1=YES, 0=NO)"
7938 INPUT L
7940 PRINT
7942 IF L=1 THEN GOTO 7878
7944 END

```

程序运行:

```

RUN
INPUT DIGREE OF POLYNOMIAL N
? 2
INPUT COEFFICIENTS A(1), ..., A(N+1), (4/LINE)
? 1? -4.01? 4.02? 0
INPUT YOUR ESTIMATE OF ROOT
? 1
ROOT IS: 2.00004
ANOTHER ROOT IS? (1=YES, 0=NO)
? 1
INPUT YOUR ESTIMATE OF ROOT
? 1.5
ROOT IS: 2.00004
ANOTHER ROOT IS? (1=YES, 0=NO)
? 1
INPUT YOUR ESTIMATE OF ROOT
? 5
ROOT IS: 2.00995
ANOTHER ROOT IS? (1=YES, 0=NO)
? 0
END AT 7944

```

## §2 半区间搜索法求多项式的实根

### 一、方法概要

设  $f(x)$  是  $X$  连续函数, 为具体起见, 设  $a < b$ ,  $f(a) < 0$ ,  $f(b) > 0$ , 则  $f(x) = 0$  在  $(a, b)$  之间一定有实根, 也就是说  $(a, b)$  区间是方程的有根区间, 用  $a$  和  $b$  的中点  $(a+b)/2$  平分

$(a, b)$  为两个区间, 计算  $f((a+b)/2)$  的函数值, 如果  $f(\frac{a+b}{2}) = 0$ , 则根为  $\frac{a+b}{2}$ ,

如果  $f(\frac{a+b}{2}) < 0$ , 则改此中点为  $a_1$ , 改  $b$  为  $b_1$ ; 如果  $f(\frac{a+b}{2}) > 0$ , 则令此中点为  $b_1$ , 改



$a$  为  $a_1$ 。在后两种情况,  $(a_1, b_1)$  是新的有根区间, 在  $(a, b)$  之内, 长度为原有根区间的一半。在后两种情况再计算  $f(\frac{a_1+b_1}{2})$ , 如果碰巧  $f(\frac{a_1+b_1}{2})=0$ , 则找到方程的根为  $\frac{a_1+b_1}{2}$ 。如果  $f(\frac{a_1+b_1}{2}) < 0$ , 则令中点  $\frac{a_1+b_1}{2}$  为  $a_2$ , 改  $b_1$  为  $b_2$ ; 如果  $f(\frac{a_1+b_1}{2}) > 0$ , 则令中点  $\frac{a_1+b_1}{2}$  为  $b_2$ , 改  $a_1$  为  $a_2$ 。在后两种情况中  $(a_2, b_2)$  又是一个新的有根区间, 在  $(a_1, b_1)$  之内, 是  $(a, b)$  的长度的  $1/4$ 。这样做  $n$  次, 如果还没有找到方程的根, 我们就有一个区间, 当  $n$  充分大时, 我们可取最后区间的中点作为方程的根的近似值  $x_n = \frac{1}{2}(a_n + b_n)$ , 它的误差小于  $\frac{b-a}{2^{n+1}}$ 。

## 二、程序说明:

1. 使用本程序, 必须在语句 7770 用 DEF FNC (X) 语句定义要求根的函数  $f(x)$ 。

例如: 函数  $f(x) = -3x^5 + x^3 + 1$ , 应如下写入:

```
7770 DEF FNC(X)=-3 * X * * 5 + X * * 3 + 1
```

2. 程序中使用数组 D
3. 程序中使用工作单元 A, B, C, S, Q, T, X, V, U, N
4. 本程序不能求重根及虚根

## 三、程序使用:

1. 用 BASIC 解释程序引入本程序, 键盘命令 RUN 启动程序执行, 机器自动印出: "INTERVAL (LOWER LIMIT, UPPER LIMIT)" 询问有根区间的下限和上限。
2. 由键盘输入有限区间上限和下限, 机器立即印出结果  
"ONE ROOT AT: \* \* \*"
3. 然后机器印出 "MORE INPUT (1=YES, 0=NO)" 询问还要求新的根吗? 若要, 回答 "1", 否则回答 "0"
4. 若回答为 "1", 重复第(2)步, 若回答 "0", 程序结束。

## 四、试题

$$f(x) = x^4 + 2x^3 - 13x^2 - 14x + 24$$

根 1, -2, 3, -4。

## 五、程序清单及运行结果:

### 程序清单

```
7770 DEF FNC(X)=X * X * X * X + 2 * X * X * X - 13 * X * X - 14 * X + 24
7772 DIM D(4)
7774 PRINT "INTERVAL (LOWER LIMIT, UPPER LIMIT)"
7776 INPUT A, B
7778 LET C=FNC(A)
7780 LET S=SGN(C)
7782 LET Q=FNC(B)
```

```

7784 LET T=SGN(Q)
7786 PRINT
7788 IF S*T=0 THEN GOTO 7832
7790 IF S*T<0 THEN GOTO 7814
7792 FOR I=1 TO 1000
7794 LET X=A+RND(1)*(B-A)
7796 LET P=FNC(X)
7798 LET V=SGN(P)
7800 IF V=0 THEN GOTO 7840
7802 IF S=<0 THEN GOTO 7812
7804 NEXT I
7806 PRINT "NOCHANGE OF BIGN FDUNG"
7808 PRINT
7810 GOTO 7842
7812 LET B=X
7814 LET D(2+S)=A
7816 LET D(2-S)=B
7818 LET X=(D(1)+D(3))/2
7820 LET U=SGN(FNC(X))
7822 IF U=0 THEN GOTO 7840
7824 LET D(2+U)=X
7826 LET Q=ABS(D(1)-D(3))/(ABS(D(1))+ABS(D(3)))
7828 IF Q<.000001 THEN GOTO 7840
7830 GOTO 7818
7832 IF S=0 THEN GOTO 7838
7834 LET X=B
7836 GOTO 7840
7838 LET X=A
7840 PRINT "ONE ROOT AT: ", X
7842 PRINT
7844 PRINT "MORE INPUT(1=YES, 0=NO)"
7846 INPUT N
7848 PRINT
7850 IF N=1 THEN GOTO 7774
7852 END

```

程序运行结果:

RUN

INTERVAL(LOWER LIMIT, UPPER LIMIT)

? -3? -1.5

ONE ROOT AT: -2

MORE INPUT (1=YES, 0=NO)

? 1

INTERVAL(LOWER LIMIT, UPPER LIMIT)

? 2? 9

ONE ROOT AT: 3

MORE INPUT(1=YES, 0=NO)

? 0

END AT 7852

### §3 BH法求多项式的实根和复根

#### 一、方法概要:

BH法是确定实系数多项式的二次因子  $x^2+px+q$  的方法, 解此二次因子, 就可以得出多项式的两个根  $x_1$  和  $x_2$ 。此方法全名叫做 Bairstow—Hitchcock法。

给定实型多项式方程

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n=0 \quad (8-3-1)$$

把 (8-3-1) 式除以实验因子  $x^2+px+q$  得:

$$f(x)=(x^2+px+q)Q_1(x)+RX+S \quad (8-3-2)$$

其中  $Q_1(x)=b_0x^{n-2}+b_1x^{n-3}+\cdots+b_{n-3}x+b_{n-2}$

而  $R, S$  分别是  $p, q$  的函数。

如果  $R=S=0$ , 则  $x^2+px+q$  就是  $f(x)$  的二次因子, 于是确定二次因子  $x^2+px+q$  的问题就归结为解联立方程:

$$\begin{cases} R(p, q)=0 \\ S(p, q)=0 \end{cases} \quad (8-3-3)$$

设 (8-3-3) 的解为  $\alpha_1$  和  $\alpha_2$ , 并设 (8-3-3) 的近似解为  $p$  和  $q$

$$\alpha_1-p=\delta_1 \quad \alpha_2-q=\delta_2$$

很明显,  $\alpha_1, \alpha_2$  以及  $\delta_1, \delta_2$  分别是  $p, q$  的函数。把  $R, S$  在  $\alpha_1$  和  $\alpha_2$  处展开成台劳级数, 并略去二次和二次以上的项, 得:

$$\begin{cases} \frac{\partial R}{\partial p}(p-\alpha_1)+\frac{\partial R}{\partial q}(q-\alpha_2)+R(p, q)=0 \\ \frac{\partial S}{\partial p}(p-\alpha_1)+\frac{\partial S}{\partial q}(q-\alpha_2)+S(p, q)=0 \end{cases}$$
$$\therefore \begin{cases} R(p, q)=\frac{\partial R}{\partial p}\delta_1(p, q)+\frac{\partial R}{\partial q}\delta_2(p, q)=R_p\delta_1+R_q\delta_2 \\ S(p, q)=\frac{\partial S}{\partial p}\delta_1(p, q)+\frac{\partial S}{\partial q}\delta_2(p, q)=S_p\delta_1+S_q\delta_2 \end{cases}$$

$$\text{如果 } D=\begin{vmatrix} R_p & R_q \\ S_p & S_q \end{vmatrix} \neq 0$$

则

$$\begin{cases} \delta_1 = \frac{1}{D} \cdot \begin{vmatrix} R_p & R(p, q) \\ S_p & S(p, q) \end{vmatrix} \\ \delta_2 = \frac{1}{D} \cdot \begin{vmatrix} R(p, q) & Rq \\ S(p, q) & Sq \end{vmatrix} \end{cases} \quad (8-3-6)$$

我们就把 $\delta_1$ 和 $\delta_2$ 分别作为 $p, q$ 向 $\alpha_1$ 和 $\alpha_2$ 逼近的增量。即：

$$\begin{aligned} p_{i+1} &= p_i + \delta_1 = p_i + \Delta p(p_i, q_i) \\ q_{i+1} &= q_i + \delta_2 = q_i + \Delta q(p_i, q_i) \end{aligned} \quad (8-3-5)$$

为了求出 $\Delta p, \Delta q$ ，必须求出 $S, R, R_p, R_q, S_p, S_q$ 等，下面推导求这些量的公式。

把式(8-3-2)中的 $Q_1(x)$ 除以 $x^2+px+q$ 得：

$$Q_1(x) = (x^2+px+q)Q_2(x) + \bar{R}x + \bar{S}$$

$$\text{其中 } Q_2(x) = C_0 x^{n-4} + C_1 x^{n-5} + \dots + C_{n-6} x + C_{n-4} \quad (8-3-6)$$

现设  $b_0 = a_0, b_1 = a_1 - pb_0$

则：

$$\left. \begin{aligned} b_K &= a_K - pb_{K-1} - qb_{K-2} \quad K=2, 3, 4, \dots, n \\ C_0 &= b_0, \quad C_1 = b_1 - pC_0 \\ C_K &= b_K - pC_{K-1} - qC_{K-2} \quad K=2, 3, 4, \dots, n-1 \end{aligned} \right\} \quad (8-3-7)$$

比较(8-3-2)式和(8-3-6)式的系数有：

$$\left. \begin{aligned} R &= b_{n-1}, \quad S = b_n + pb_{n-1} \\ \bar{R} &= C_{n-3}, \quad \bar{S} = C_{n-2} + pC_{n-3} \end{aligned} \right\} \quad (8-3-8)$$

其中的 $b_n, b_{n-1}, C_{n-2}, C_{n-3}$ 都是由(8-3-7)式导出。

将(8-3-6)式两边乘以 $x$ ，得：

$$\begin{aligned} xQ_1(x) &= (x^2+px+q)Q_2(x) \cdot x + \bar{R}x^2 + \bar{S}x \\ &= (x^2+px+q)(C_0 x^{n-3} + C_1 x^{n-4} + \dots + C_{n-4}x) \\ &\quad + C_{n-3}x^2 + C_{n-2}x + pC_{n-3}x + C_{n-3}q - C_{n-3}q \\ &= (x^2+px+q) \cdot x \cdot Q_2(x) + C_{n-3}(x^2+px+q) + \\ &\quad C_{n-2}x - C_{n-3}q \end{aligned}$$

∴ 如将 $xQ_1(x)$ 除以 $(x^2+px+q)$ 所得的余式为

$$R^*x + S^*$$

则有：

$$\begin{cases} R^* = C_{n-2} \\ S^* = -C_{n-3}q \end{cases}$$

使用(8-3-7)式中最后一个关系式，令 $K=n-1$ 则有：

$$C_{n-1} = b_{n-1} - pC_{n-2} - qC_{n-3}$$

$$\therefore -qC_{n-3} = C_{n-1} - b_{n-1} + pC_{n-2}$$

所以导出：

$$\left. \begin{aligned} R^* &= C_{n-2} \\ S^* &= C_{n-1} - b_{n-1} + pC_{n-2} \end{aligned} \right\} \quad (8-3-9)$$

另一方面将(8-3-2)式两边分别对 $p, q$ 求偏导数得：

(8-3-2)式对 $p$ 求偏导数：

$$0 = xQ_1(x) + (x^2 + px + q) \frac{\partial Q_1}{\partial p} + x \frac{\partial R}{\partial p} + \frac{\partial S}{\partial p}$$

(8-3-2)式对q求偏导数:

$$0 = Q_1(x) + (x^2 + px + q) \frac{\partial Q_1}{\partial q} + x \frac{\partial R}{\partial q} + \frac{\partial S}{\partial q}$$

整理:

$$\left. \begin{aligned} xQ_1(x) &= -(x^2 + px + q) \frac{\partial Q_1}{\partial p} - x \frac{\partial R}{\partial p} - \frac{\partial S}{\partial p} \\ Q_1(x) &= -(x^2 + px + q) \frac{\partial Q_1}{\partial q} - x \frac{\partial R}{\partial q} - \frac{\partial S}{\partial q} \end{aligned} \right\} \quad (8-3-10)$$

由式(8-3-10)可以看出 $xQ_1(x)$ 和 $Q_1(x)$ 除以 $(x^2 + px + q)$ 的余式分别为:

$$-x \frac{\partial R}{\partial p} - \frac{\partial S}{\partial p} \text{ 和 } -x \frac{\partial R}{\partial q} - \frac{\partial S}{\partial q}$$

比较(8-3-8)、(8-3-9)、(8-3-10)可以得出:

$$\left. \begin{aligned} \frac{\partial R}{\partial p} &= R_p = -R^* = -C_{n-2} \\ \frac{\partial S}{\partial p} &= S_p = -S^* = -(C_{n-1} - b_{n-1} + pC_{n-2}) \\ \frac{\partial R}{\partial q} &= R_q = -\bar{R} = -C_{n-3} \\ \frac{\partial S}{\partial q} &= S_q = -\bar{S} = -(C_{n-2} + pC_{n-3}) \end{aligned} \right\} \quad (8-3-11)$$

综上所述, BH法的迭代公式为:

从 $p_0=1, q_0=1$ 出发

$$\begin{cases} p_{i+1} = p_i + \Delta p \\ q_{i+1} = q_i + \Delta q \\ \Delta p = (b_{n-1}C_{n-2} - b_nC_{n-3})/D_1 \\ \Delta q = (b_nC_{n-2} - b_{n-1}\bar{C}_{n-1})/D_1 \end{cases}$$

其中

$$\begin{aligned} D_1 &= C_{n-2}^2 - \bar{C}_{n-1}C_{n-3} \\ \bar{C}_{n-1} &= C_{n-1} - b_{n-1} \end{aligned}$$

当迭代进行到 $|\Delta p| < \varepsilon, |\Delta q| < \varepsilon$ ,  $\varepsilon$ 是给定的任意小量, (本程序 $\varepsilon=0.001$ ), 就把这时的 $p_{i+1}$ 作 $p$ ,  $q_{i+1}$ 作 $q$ , 解二次因子 $x^2 + px + q$ , 得到 $f(x)$ 的两个根, 然后将 $n-2$ , 重复以上步骤, 直到 $n=0$ 为止。

在迭代中, 规定 $D \neq 0$ 。如果 $D=0$ , 则把 $p_i, q_i$ 分别增加一个小量 $\Delta h$  (本程序 $\Delta h=0.0001$ ), 再用(8-3-12)式进行迭代。

## 二、程序说明

本程序使用的工作单元

A(N+4)输入多项式的各系数。

B(N+4), C(N+4)运算中间数组。

N, I, P, Q, M, J, U<sub>1</sub>, U<sub>2</sub>, V<sub>1</sub>, V<sub>2</sub>, V<sub>3</sub>, W<sub>1</sub>, W<sub>2</sub>, W<sub>3</sub>, S, T等皆为运行过程中的简单变量

### 三、上机操作步骤:

1. 用BASIC解释程序将本程序送入机内
2. 用键盘命令RUN启动本程序。电传机打印出:  
"INPUT DIGREE OF POLYNOMIAL N" 用户根据多项式的阶数回答。回答数字后按回车键。
3. 电传又打印出:  
"INPUT COEFF. A(0), A(1), ..., A(N) (4/LINE)"  
用户逐项将系数送入机内, 有缺项的用零补上。每送入一个数, 要按一下回车键。每行输入四个数。
4. 经过计算求出根, 并打印出所有的根, 以"ROOTS"为标志。

### 四、试题

(1)  $X^6 + X^5 - 45X^4 + 35X^3 + 524X^2 - 1236X + 720 = 0$

根为: 1, 2, 3, 4, -5, -6。

(2)  $X^8 - 30X^6 + 273X^4 - 820X^2 + 576 = 0$

根为:  $\pm 1$ ,  $\pm 2$ ,  $\pm 3$ ,  $\pm 4$ 。

(3)  $X^6 + X^4 - 8X^3 - 16X^2 + 7X + 15 = 0$

根为: 3,  $-2 \pm i$ ,  $\pm 1$ 。

(4)  $X^6 - 2X^5 + 2X^4 + X^3 + 6X^2 - 6X + 8 = 0$

根为:  $-1 \pm i$ ,  $1.5 \pm 1.322875654i$ ,  $0.5 \pm 0.8660254036i$

### 五、程序清单及运行结果

#### 程序清单

```
7620 PRINT "INPUT DIGREE OF POLYNOMIAL N"
7622 INPUT N
7624 PRINT
7626 DIM A(N+4), B(N+4), C(N+4)
7628 PRINT "INPUT COEFF. A(0), A(1), ...A(N) (4/LINE)"
7630 FOR I=0 TO INT ((N+4)/4)-1
7632   INPUT A(4*I), A(4*I+1), A(4*I+2), A(4*I+3)
7634   PRINT
7636 NEXT I
7638 PRINT "ROOTS"
7640 LET L=0
7642 IF N>1 THEN GOTO 7648
7644 PRINT -A(1)/A(0),
7646 GOTO 7768
7648 IF N>2 THEN GOTO 7658
7650 LET P=A(1)/A(0)
7652 LET Q=A(2)/A(0)
7654 LET L=1
7656 GOTO 7726
```

```

7658 LET P=1
7660 LET Q=1
7662 LET M=1
7664 LET B(0)=A(0)
7666 LET C(0)=A(0)
7668 LET B(1)=A(1)-P*B(0)
7670 LET C(1)=B(1)-P*C(0)
7672 FOR J=2 TO N
7674   LET B(J)=A(J)-P*B(J-1)-Q*B(J-2)
7676   LET C(J)=B(J)-P*C(J-1)-Q*C(J-2)
7678 NEXT J
7680 LET U1=B(N)
7682 LET U2=B(N-1)
7684 LET V1=C(N-1)-U2
7686 LET V2=C(N-2)
7688 LET V3=C(N-3)
7690 LET W1=V2*V2-V1*V3
7692 IF ABS(W1)>.1E-10 THEN GOTO 7700
7694 LET P=P+.0001
7696 LET Q=Q+.0001
7698 GOTO 7668
7700 LET W2=(U2*V2-U1*V3)/W1
7702 LET W3=(U1*V2-U2*V1)/W1
7704 LET P=P+W2
7706 LET Q=Q+W3
7708 IF ABS(W2)>.001 THEN GOTO 7714
7710 IF ABS(W3)>.001 THEN GOTO 7714
7712 GOTO 7720
7714 IF M>1000 THEN GOTO 7766
7716 LET M=M+1
7718 GOTO 7668
7720 FOR J=0 TO N-2
7722   LET A(J)=B(J)
7724 NEXT J
7726 LET P=P/2
7728 LET T=P*P-Q
7730 IF T=>0 THEN GOTO 7738
7732 LET S=SQR(-T)
7734 PRINT -P, "+I" S, -P, "-I", S,
7736 GOTO 7748

```

```

7738 LET S=SQR(T)
7740 IF P>0 THEN GOTO 7746
7742 PRINT S-P, Q/(S-P),
7744 GOTO 7748
7746 PRINT -P-S, Q/(-P-S),
7748 LET B(N)=W2
7750 LET C(N)=W3
7752 LET B(N-1)=U2
7754 LET C(N-1)=V1
7756 IF L=1 THEN GOTO 7768
7758 LET N=N-2
7760 IF N >2.5 THEN GOTO 7658
7762 IF N >1.5 THEN GOTO 7648
7764 GOTO 7642
7766 PRINT "FINAL"
7768 END

```

运行结果

(1) 第一试题运行结果

RUN

INPUT DIGREE OF POLYNOMIAL N

? 6

INPUT COEFF. A(0), A(1), ...A(N)(4/LINE)

? 1? 1? -45? 35

? 524? -1236? 20? 0

ROOTS

3.99999	1	-5	2.00038
---------	---	----	---------

- 6	2.99968
-----	---------

END AT 7768

(2) 第三试题运行结果

RUN

INPUT DIGREE OF POLYNOMIAL N

? 5

INPUT COEFF. A(0), A(1), ...A(N) (4/LINE)

? 1? 1? -8? -16

? 7? 15? 0? 0

ROOTS

- 1	1	-2.00001+I	1.00002
-----	---	------------	---------

-2.00001	-I	1.00002	2.99999
----------	----	---------	---------

END AT 7768

(3) 第二试题运行结果



RUN

INPUT DIGREE OF POLYNOMIAL N

? 8

INPUT COEFF. A(0), A(1), ...A(N) (4/LINE)

? 1? 0? -30? 0

? 273? 0? -820? 0

? 576? 0? 0? 0

ROOTS

1	- 1	-2.00004	1.99999
3.00002	-2.99998	- 4	4

END AT 7768

(4) 第四试题运行结果

RUN

INPUT DIGREE OF POLYNOMIAL N

? 6

INPUT COEFF. A(0), A(1), ...A(N) (4/LINE)

? 1? -2? 2? 1

? 6? -6? 8? 0

ROOTS

- 1 + I 1	- 1 - I 1	.499995 + I .865771
.499995 - I .865771		1.49999 + I 1.323
1.49999 - I 1.323		

END AT 7768

#### 本章参考文献

- (1) 《常用算法》 科学出版社 1976
- (2) 《计算技术通讯》 1975年1—2期

## 第九章 求 定 积 分

### §1 辛普生 (Simpson) 求定积分

#### 一、方法概要:

若 $f(x)$ 为已知确定函数, 则辛普生求积公式为:

$$\int_A^B f(x)dx = \frac{D}{3} \{f(A) + 4f(A+D) + 2f(A+2D) + 4f(A+3D) + 2f(A+4D) + \dots + f(B)\}$$

若 $f(x)$ 为未知的确定函数, 只知各个等距点的函数值, 则辛普生求积公式为:

$$\int_A^B f(x)dx = \frac{D}{3} (y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \dots + 2y_{n-2} + 4y_{n-1} + y_n)$$

其中 A——积分下限

B——积分上限

D——x 的增量,

$$D = \frac{B-A}{2n}$$

#### 二、程序说明

1. 程序未使用数组。

2. 程序使用简单变量为 Z, A, B, D, Y<sub>1</sub>, Y, Y<sub>2</sub>, S, T

3. 当 Z=1 表示  $f(x)$  为已知确定显函数, 使用本程序前, 除在标号 7646 语句中用 DATA 语句依次给出 Z, A, B, D 值外, 还须在标号 7600 语句中用语句 DEF FNC(x) 定义  $f(x)$  函数。

当 Z=0 表示  $f(x)$  为已知的隐函数。使用本程序前, 应在标号 7646 语句中用 DATA 语句依次给出 Z, A, B, D 及  $f(x)$  的各个等距点的函数值。

#### 三、操作说明

用 BASIC 解释程序引入本程序, 键盘命令 RUN ↵ 启动本程序, 机器立即印出结果,

INTEGRAT=\*\*\* 程序结束。

#### 四、试题

1. Z=1  $f(x)=x^3+2x^2+3x+4$

A=1, B=2, D=0.1

$$\int_1^2 f(x)dx = 16.9166$$

2. Z=0 A=1, B=2, D=0.1

Y<sub>1</sub>=1, 1.331, 1.728, 2.197, 2.744, 3.375,  
4.096, 4.913, 5.832, 6.859, 8

$$\int_1^2 f(x)dx = 3.75$$

## 五、程序清单及运行结果

### 程序清单

```
7600 DEF FNC(X)=X*X*X+2*X*X+3*X+4
7602 READ Z, A, B, D
7604 LET S=0
7606 LET T=0
7608 PRINT
7610 IF Z=1 THEN GOTO 7616
7612 READ Y1
7614 GOTO 7620
7616 LET Y1=FNC(A)
7618 LET Y2=FNC(B)
7620 FOR I=1 TO (B-A)/D+.5
7622 IF Z=1 THEN GOTO 7628
7624 READ Y
7626 GOTO 7630
7628 LET Y=FNC(A+I*D)
7630 IF I/2=INT(I/2) THEN GOTO 7636
7632 LET S=S+Y
7634 GOTO 7638
7636 LET T=T+Y
7638 NEXT I
7640 IF Z=1 THEN GOTO 7644
7642 READ Y2
7644 PRINT "INTEGRAT,"; (D/3)*(Y1+4*S+2*T+Y2)
7646 DATA 1,2,.1
7648 DATA 1,1.331,1.728,2.197,2.744,3.375
7650 DATA 4.096,4.913,5.832,6.859,8
7652 END
```

运行结果:

RUN

INTEGRAT; 16.9167

END AT 7652

## §2 高斯法求定积分

### 一、方法概要

给定积分限A, B和被积函数f(x), 用K个结点的高斯-勒让特法求积公式可求出积分

$$I = \int_A^B f(x)dx = \sum_{i=1}^K V_i f(X_i)$$

在结点固定的情况下,它具有最高代数精确度 $2K-1$

其中 $X_i$ 是结点,  $V_i$ 是相应结点 $X_i$ 的权系数, 它们都是预先给定的, 对任何函数都一样。

$f(x)$ 是被积函数。

积分时将 $(A, B)$ 区间分成 $K$ 个子区间。在每个子区间上应用高斯-勒让积分公式求出 $f(x)$ 在各子区间上的积分。最后将这 $K$ 个积分相加即得 $f(x)$ 在 $(A, B)$ 区间上的积分。

## 二、程序说明

1. 程序没有使用数组
2. 程序使用简单变量 $A, B, K, C, D, T, J, S, I, X, N$
3. 被积函数在语句7660中用自定义函数给出, 结点 $X_i$ 和相应权系数 $V_i$ 在语句7720以后已给出, 而子区间数由键盘输入。程序结束积分结果在 $I$ 中。

三、上机操作说明: 以下题为例, 说明如何使用本程序。

$$f(x) = x^2 \cos(x)$$

$$\int_0^3 f(x) dx$$

1. 由BASIC解释程序将本程序引入机器内

在语句7660中定义函数

$$7660 \text{ DEF FNC}(X) = X * X * \text{COS}(X)$$

2. 键盘命令RUN, 启动本程序, 机器自动印出:

"ENTER LOWER LIMIT OF INTEGRATION A UPPER LIMIT OF INTEGRATION B", 询问积分下限和上限。

3. 由键盘输入积分下限和上限, 电传机又印出: "ENTER NO. OF SUBINTERVALS"

要求输入子区间的个数 $K$ 。

4. 由键盘输入子区间的个数, 电传机立即印出计算的结果:

$$\text{INTERAL} = * * *$$

并接着印出: "CHANGE NUMBER OF SUBINTERVALS (1=YES, 0=NO)" 询问要改变子区间的个数 $K$ 吗?

若要, 回答"1", 机器印出"ENTER NO. OF SUBINTERVALS"重复步骤(4);

否则回答"0", 机器印出: "NEW INTEGRATION LIMITS? (1=YES 0=NO)" 询问要改变积分限吗? 如果有新的积分限, 则由键盘回答"1", 程序将转到

步骤(2), 询问新的积分下限和上限, 进行积分运算。如果没有新的积分限则由键盘输入"0", 程序结束。

## 四、试题

$$f(x) = x^2 \cos x$$

$$\int_0^3 f(x) dx$$

当子区间个数为"8"时, 积分值 $I = -4.94705$

## 五、程序清单及运行结果

程序清单

```

7660 DEF FNC(X)=X↑2 * COS(X)
7662 PRINT "ENTER LOWIT LIMIT OF INTEGRATION A"
7664 PRINT "UPPER LIMIT OF INTERGRAGON B"
7666 INPUT A, B
7668 PRINT
7670 PRINT "ENTER NO. OF SUBINTERVALS"
7672 INPUT K
7674 PRINT
7676 LET C=(B-A)/K/2
7678 LET D=A+C
7680 LET T=0
7682 FOR J=1 TO K
7683   LET S=0
7684   FOR I=1 TO 10
7686     READ X,N
7688     LET S=S+N*(FNC(C*X+D)+FNC(D-C*X))
7690   NEXT I
7692   RESTORE
7694   LET T=T+S*C
7696   LET D=D+2*C
7698 NEXT J
7700 PRINT "INTEGRAL=",T
7702 PRINT
7704 PRINT "CHANGE NUMBER OF SUBINTERVALS(1=YES, 0=NO)"
7706 INPUT I
7708 PRINT
7710 IF I=1 THEN GOTO 7670
7712 PRINT "NEW INTEGRATION LIMIT(1=YES, 0=NO)"
7714 INPUT I
7716 PRINT
7718 IF I=1 THEN GOTO 7662
7720 DATA 7.65265E-2,.152753,.227786,.149173
7722 DATA .373706,.142096,.510867,.131689
7724 DATA .636054,.118194,.746331,.101193
7726 DATA .839117,8.32769E-2,.812234,.062672
7728 DATA .963972,.0406,.99,.017614,.993129,1.7614E-4
7730 END

```

运行结果:

RUN

ENTER LOWIT LIMIT OF INTEGRATION A

UPPER LIMIT OF INTERGRAGION B

? 0 ? 3

ENTER NO. OF SUBINTERVALS

? 8

INTEGRAL=-4.94705

CHANGE NUMBER OF SUBINTERVALS (1=YES, 0=NO)

? 0

NEW INTEGRATION LIMIT (1=YES, 0=NO)

? 0

END AT 7740

### §3 梯形法求定积分

#### 一、方法概要

若 $f(x)$ 为已知确定的显函数, 则梯形法求定积分的公式为:

$$\int_A^B f(x)dx = \sum_{i=0}^{N-1} \{f(A+H \times i) + f(A+H \times (i+1))\} \times H/2$$

若 $f(x)$ 为已知的隐函数, 即只知各个等距点的函数值, 则梯形法求定积分公式为:

$$\int_A^B f(x)dx = \sum_{i=0}^{N-1} (y_i + y_{i+1}) \times H/2$$

其中:

A——积分下限

B——积分上限

N——采样点的个数

H——X的增量。H=(B-A)/N

#### 二、程序说明

1. 程序使用数组Y。

2. 程序使用简单变量: Z, N, A, B, I, T, S, H, M, X。

3. Z=0表示 $f(x)$ 为已知确定的函数。使用本程序前除将Z赋值为“0”外, 还须在标号7030语句中用语句DEF FNA(x)定义 $f(x)$ 函数。

Z=1表示 $f(x)$ 为已知的隐函数, 在标号7052以后的语句中用DATA语句依次给出 $f(x)$ 的各个等距点上的函数值。

#### 三、操作说明

1. 用BASIC解释程序引入本程序。键盘命令RUN启动。机器立即印出: “Z=, N=, A=, B=” 询问是显函数还是隐函数; 采样点的个数N; 积分下限A; 积分上限B。

2. 当键盘输入Z, N, A, B后机器立即印出结果:

S=\*\*\*

3. 印出积分结果后, 机器又印出

“MORE INPUT (1=YES, 0=NO)”

如果回答“1”，机器就转回到开始，询问新的积分限等。如果回答“0”，程序结束。

#### 四、试题

1.  $Z=0$ ,  $N=10$ ,  $A=0$ ,  $B=3.1415926$

$$f(x)=\sin(x)$$

$$S = \int_0^{3.1415926} f(x)dx = 1.98352$$

2.  $Z=0$ ,  $N=10$ ,  $A=0$ ,  $B=6.28318$

$$f(x)=\sin(x)$$

$$S = \int_0^{6.28318} f(x)dx = -5.36442 \times 10^{-7}$$

3.  $Z=1$ ,  $N=10$ ,  $A=0$ ,  $B=3.1415926$  采样点为: 0, 0.309, 0.587, 0.809, 0.951, 1, 0.951, 0.809, 0.587, 0.31, 0.

$$S = 1.98329$$

#### 五、程序清单及运行结果

##### 程序清单

```
7000 DIM Y(200)
7002 LET S=0
7004 PRINT "Z=",N,"A=",B="
7006 INPUT Z,N,A,B
7008 PRINT
7010 LET H=(B-A)/N
7012 IF Z=0 THEN GOTO 7030
7014 FOR I=1 TO N+1
7016 READ Y(I)
7018 NEXT I
7020 FOR I=1 TO N
7022 LET T=(Y(I)+Y(I+1))*H/2
7024 LET S=S+T
7026 NEXT I
7028 GOTO 7040
7030 DEF FNA(X)=SIN(X)
7032 FOR I=0 TO N-1
7034 LET T=(FNA(A+H*I)+FNA(A+H*(I+1)))*H/2
7036 LET S=S+T
7038 NEXT I
7040 PRINT "S=", S
7042 PRINT "MORE INPUT (1=YES, 0=NO)"
7044 INPUT M
7046 PRINT
```

```

7048 IF M=1 THEN GOTO 7002
7050 END
7052 DATA 0,.309,.587,.809,.951,1
7054 DATA .951,.809,.587,.31,0

```

运行

```

RUN
Z=, N=, A=, B=
? 0? 10? 0? 3.1415926
S=1.98352
MORE INPUT (1=YES, 0=NO)
? 1
Z=, N=, A=, B=
? 0? 10? 0? 6.28318
S=-5.36442E-7
MORE INPUT (1=YES, 0=NO)
? 1
Z=, N=, A=, B=
? 1? 10? 0? 3.1415926
S=1.98329
MORE INPUT (1=YES, 0=NO)
? 0
END AT 7050

```

### 参 考 资 料

1. 《电子计算机常用算》

科学出版社

2. 《计算方法》

武汉大学、山东大学

人民教育出版社

3. 《计算方法》 北京大学、山东大学

科学出版社



## 第十章 微分及龙格-库塔法求解一阶微分方程组

### §1 微 分

#### 一、方法概要

本程序是计算任给函数 $f(x)$ 在点 $X=a$ 处的微分值，这个值的确定是由逐次逼近 $X$ 的正确的差分商。

$$\text{差分商} = \frac{F(X) - F(a)}{X - a}$$

其中  $a$ ——要求微分的已知点的值

$$x = a + 0.5^n$$

$n=1-10$ 的循环值

所以，在  $a$  点的微分  $= 2 * (\text{第}M\text{次的循环差分商}) - (\text{第}M-1\text{次循环差分商})$

#### 二、程序说明

##### 1. 本程序使用的工作单元

D 第 $M$ 次循环为差分商

X 中间变量

A 求微分的已知点的值

D, 第 $M-1$ 次循环差分商

2. 一函数的微分值决定于函数形式，因为函数形式是不固定的，用户在使用本程序时，应根据情况修改6002语句，以适应用户的不同函数。

#### 三、上机操作步骤

1. 用BASIC解释程序将本程序输入机内。

2. 用键盘命令RUN↵，使本程序投入运行。电传打印出：

"ENTER VALUE OF A (TO END PROG. INPUT 99999)"

3. 用户根据要计算的微分值，给出已知点A的值。（如果要使本程序结束运行，则输入99999，并回车，本程序即行结束），并给出求差分商的循环次数M。

4. 电传又打印出：

"X DIFFERENCE QUOTIENT"

× × × × × ×

...

× × × × × ×

打印出每一个X值，以及相应的差分商值。

5. 电传最后打印出：

"DERIVATION AT X = (a) IS: × × ×"

6. 打印结束后，又要求用户输入A点的值，电传打印出：

"ENTER VALVE OF A"

重新返回到第三步运行。

#### 四、例题

已知  $f(x) = x^5 + x^3 + x - 5$

则  $f'(1) = 9$

#### 五、程序清单及运行结果

##### 程序清单

```
6000 PRINT "ENTER VALUE OF A (TO END PROG. INPUT 99999) "  
6002 DEF FNC (x) =X↑ 5 +X↑ 3+X-5  
6004 INPUT A  
6006 IF A=99999 THEN GOTO 6038  
6008 INPUT M  
6010 PRINT  
6012 PRINT "X", "DIFFERENCE QUOTIENT"  
6014 LET D= 0  
6016 FOR N=1 TO M  
6018 LET D1=D  
6020 LET X=A+.5↑N  
6022 LET D=(FNC(X)-FNC(A))/(X-A)  
6024 PRINT INT (X*10000+.5) /10000, D  
6026 NEXT N  
6028 PRINT  
6030 PRINT "DERIVATIVE AT X="; A, "IS:"; 2*D-D1  
6032 PRINT  
6034 PRINT "ENTER VALUE OF A"  
6036 GOTO 6004  
6038 END
```

##### 四、程序运行

RUN

ENTER VALUE OF A (TO END PROG. INPUT 99999)

? 1 ? 10

X	DIFFERENCE QUOTIENT
---	---------------------

1.5	18.9375
1.25	13.0195
1.125	10.8069
1.0625	9.8567
1.0313	9.41714
1.0156	9.20581
1.0078	9.10229
1.0039	9.05078
1.002	9.02539

1.001                      9.0127  
 DERIVATIVE AT X=i IS: 9  
 ENTER VALUE OF A  
 ? 99999  
 END AT 6038

## §2 龙格-库塔法解一阶微分方程组

### 一、计算方法概要

本程序是四阶龙格库塔法，它的基本思想和理论可在二阶龙格库塔法公式的推导中看出来。下面讲一讲二阶龙格库塔法的推导

考虑一个常微分方程，

$$\frac{dy}{dt} = f(t, y) \quad (10-2-1)$$

当  $t=t_0$  时，初值  $y=y^0$ ，求  $t_1=t_0+h$  时的解  $y(t_1)$ ，用台劳展开式在  $(t_0, y^0)$  展开得：

$$y' = y(t_1) = y(t_0+h) = y^0 + \frac{dy}{dt} \Big|_{t=t_0, y=y^0} \cdot h + \frac{h^2}{2} \cdot \frac{d^2y}{dt^2} \Big|_{t=t_0, y=y^0} + \dots \quad (10-2-2)$$

用 (10-2-1) 式代入 (10-2-2) 式有：

$$\frac{dy}{dt} \Big|_{t=t_0, y=y^0} = f(t_0, y^0)$$

$$\frac{d^2y}{dt^2} \Big|_{t=t_0, y=y^0} = \frac{d}{dt} \left( \frac{dy}{dt} \right) \Big|_{t=t_0, y=y^0} = \left[ \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} \frac{dy}{dt} \right] \Big|_{t=t_0, y=y^0} = \frac{\partial f}{\partial t} \Big|_{t=t_0, y=y^0} + f(t_0, y^0) \frac{\partial f}{\partial y} \Big|_{t=t_0, y=y^0}$$

所以 (10-2-2) 式可得：

$$y' = y(t_1) = y(t_0+h) = y^0 + f(t_0, y^0)h + \frac{h^2}{2} \left( \frac{\partial f}{\partial t} \Big|_{t=t_0, y=y^0} + f(t_0, y^0) \frac{\partial f}{\partial y} \Big|_{t=t_0, y=y^0} \right) + \dots \quad (10-2-3)$$

另一方面，假设这个解可以写成：

$$y' = y^0 + (a_1 K_1 + a_2 K_2)h \quad (10-2-4)$$

其中 令  $K_1 = f(t_0, y^0)$

$$K_2 = f(t_0 + b_1 h, y^0 + b_2 K_1 h)$$

$a_1, a_2, b_1, b_2$  是四个待定常数

将  $K_1, K_2$  代入 (10-2-4)，并将  $K_2$  在  $(t_0, y^0)$  点用台劳展开式展开得：

$$y' = y^0 + a_1 f(t_0, y^0)h + [a_2 f(t_0 + b_1 h, y^0 + b_2 f(t_0, y^0)h)]h$$

$$\therefore f(t_0 + \Delta t, y^0 + \Delta y) = f(t_0, y^0) + \Delta t \frac{\partial f}{\partial t} \Big|_{t=t_0, y=y^0} + \Delta y \frac{\partial f}{\partial y} \Big|_{t=t_0, y=y^0}$$

$$\begin{aligned}\text{其中 } \Delta t &= b_1 h \\ \Delta y &= b_2 f(t_0, y^0) h\end{aligned}$$

$$\begin{aligned}\therefore y' &= y^0 + a_1 f(t_0, y^0) h + a_2 f(t_0, y^0) h + \left[ a_2 b_1 h \frac{\partial f}{\partial t} \right]_{t=t_0, y=y^0} + \\ &\quad \left[ a_2 b_2 f(t_0, y^0) h \frac{\partial f}{\partial y} \right]_{t=t_0, y=y^0} h \quad (10-2-5)\end{aligned}$$

将 (10-2-5) 和 (10-2-3) 对比确定  $a_1, a_2, b_1, b_2$  有:

$$\begin{cases} a_1 + a_2 = 1 \\ a_2 b_1 = \frac{1}{2} \\ a_2 b_2 = \frac{1}{2} \end{cases}$$

$\because$  方程数目少于未知数, 可自由选取  $b_1 = b_2 = 1$ , 于是

$$a_1 = a_2 = \frac{1}{2}$$

由此得出二阶龙格库塔法的计算公式:

$$\begin{aligned}y' &= y^0 + \frac{h}{2} (K_1 + K_2) \\ K_1 &= f(t_0, y^0) \\ K_2 &= f(t_0 + h, y^0 + K_1 h)\end{aligned}$$

可见只要利用微分方程的右端函数  $f$ , 就可以求出系数  $K_1, K_2$  得出  $y^1$ ; 再以  $y^1$  作初值, 套用以上公式求出  $y^2, \dots$ , 这样一步一步地计算到某个给定的  $t = t_{\max}$  为止。每一步把方程的解展开时, 都舍去了  $h^3$  及更高次的项, 通常说二阶龙格库塔法的局部 (即在一点) 逼近误差比例于  $h^3$ 。

完全依照前面的作法, 经过繁锁得多的推导, 可以求出四阶龙格库塔法的公式, 其局部逼近误差比例于  $h^5$ 。由于待定系数多于方程数目, 系数取法不一, 常见的一种方案是:

$$y^1 = y^0 + \frac{h}{6} (K_1 + 2K_2 + 2K_3 + K_4) \quad (10-2-6)$$

$$K_1 = f(t_0, y^0) \quad (10-2-7)$$

$$K_2 = f(t_0 + \frac{h}{2}, y^0 + \frac{h}{2} K_1) \quad (10-2-8)$$

$$K_3 = f(t_0 + \frac{h}{2}, y^0 + \frac{h}{2} K_2) \quad (10-2-9)$$

$$K_4 = f(t_0 + h, y^0 + h K_3) \quad (10-2-10)$$

本程序用的是四阶龙格库塔法的公式,

## 二、程序说明

### 1. 本程序使用的工作单元

X (I)      初始状态  
N          方程的数目

T, D, B 分别为时间初值, 步长和终值

K (4, N) 四阶龙格库塔法的四个系数

2. 本程序在对不同的一阶微分方程组求解过程中, 方程组求值也应不同. 这时应在本程序中修改6138~6142语句, 也就是子程序部分对不同的方程组应改写成不同的内容。
3. 本程序可整个作为一个子程序使用。

### 三、上机操作说明

1. 用BASIC解释程序将本程序送入计算机内。
2. 在键盘上输入命令RUN↵, 启动本程序执行。这时在电传上即打印出:  
"INPUT NUMBER OF DIFFERENTIAL EQUATIONS N"
3. 用户根据需要输入方程组的方程个数, 以回车键结束, 这时, 电传又打印出:  
"INPUT INITIAL VALUE T, EXTREME VALUE B AND STEP OF D"
4. 用户根据要求输入自变量区间初始值T和终止值B以及步长D。每输入一个数需按一次回车键。然后, 电传又打印出:  
"INPUT INITIAL VALUE OF X (I) "
5. 用户根据需要输入因变量X (I) 的初始值, 逐个输入, 每输入一个数需按一次回车键。
6. 最后计算机将计算结果打印出来。每一步打印一个结果。

### 四、例题

$$\begin{cases} \frac{dx_1}{dt} = x_2 \\ \frac{dx_2}{dt} = -0.1x_2 - 3.2^2 * x \end{cases}$$

### 五、程序清单及运行结果

#### 程序清单

```
6040 REM COMPUTING THE ONE ORDER DIFFERENTIAL SYSTEM
6042 REM IN RUNGE-KUTTA METHOD
6044 PRINT "INPUT NUMBER OF DIFFERENTIAL EQUATIONS N"
6046 INPUT N
6048 PRINT
6050 DIM X(2 * N), F(N), K(4, N)
6052 PRINT "INPUT INITIAL VALUE T, EXTREME VALUE B AND STEP
D"
6054 INPUT T, B, D
6056 PRINT
6058 PRINT "INPUT INITIAL VALUE OF X(I)"
6060 FOR I=1 TO N
6062 INPUT X (I)
6064 NEXT I
6066 PRINT
6068 PRINT "T",
```

```

6070 FOR I=1 TO N
6072 PRINT "X", I,
6074 NEXT I
6076 LET A=T
6078 FOR J=A TO B STEP D
6080 PRINT
6082 PRINT T,
6084 FOR I=1 TO N
6086 PRINT X(I) ,
6088 NEXT I
6090 FOR I=1 TO N
6092 LET X(N+I)=X(I)
6094 NEXT I
6096 GOSUB 6138
6098 FOR I=1 TO N
6100 LET K(1, I) =F(I)
6102 NEXT I
6104 LET D1=D/2
6106 LET T=T+D1
6108 FOR L=1 TO 3
6110 LET T=T+INT(L/3) * D1
6112 FOR I=1 TO N
6114 LET X (I) =X(N+I)+D1 * K(L, I) * INT(L/3+1)
6116 NEXT I
6118 COSUB 6138
6120 FOR I=1 TO N
6122 LET K(L+1, I)=F(I)
6124 NEXT I
6126 NEXT L
6128 FOR I=1 TO N
6130 LET X(I)=X(N+I)+D1/3 * (K(1,I)+2 * K(2,I)+2 * K(3,I)+K (4, I))
6132 NEXT I
6134 NEXT J
6136 END
6138 LET F(1)=X(2)
6140 LET F(2) =- .1 * X(2)-3.2 ↑ 2 * X(1)
6142 RETURN

```

程序运行结果:

RUN

INPUT NUMBER OF DIFFERENTIAL EQUATIONS N

? 2

INPUT INITIAL VALUE T, EXTREME VALUE B AND STEP D

? 0 ? 1 ? , 2

INPUT INITIAL VALUE OF X (I)

? 1 ? 0

T	X 1	X 2
0	1	0
.2	.803549	-1.88924
.4	.297132	-3.00134
.6	-.314978	-2.91771
.8	-.791407	-1.69562
1	-.94877	.163931

END AT 6136

**本章参考文献:**

南京大学数学系计算数学专业编

《常微分方程数值解法》 科学出版社 1979

# 第十一章 插 值

## §1 线 性 内 插 法

### 一、方法概要

本程序用于在一条直线上线性插入若干点, 给定 $X$ , 求 $Y$ 。(需已知此直线上两点的坐标 $X_1, Y_1$ 和 $X_2, Y_2$ )。

$$\text{由于 } \frac{Y-Y_1}{X-X_1} = \frac{Y_2-Y_1}{X_2-X_1}$$

$$Y-Y_1 = \frac{(Y_2-Y_1)(X-X_1)}{(X_2-X_1)}$$

$$Y = Y_1 + \frac{(Y_2-Y_1)(X-X_1)}{(X_2-X_1)}$$

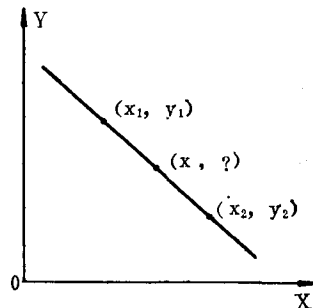


图 11-1-1

### 二、程序说明

本程序使用工作单元 $X_1, Y_1, X_2, Y_2, X, Y$

### 三、操作说明

1. 由BASIC解释程序将本程序输入机内。
2. 键盘命令RUN启动本程序执行, 电传印出:  
"INPUT KNOWN POINTS ( $X_1, Y_1, X_2, Y_2$ )"
3. 键盘输入已知二点坐标 $X_1, Y_1, X_2, Y_2$ 并再询问插入点 $X$ 。
4. 键盘输入 $X$ 值, 电传打出该插入点之 $Y$ 值, 并再问 $X$ 值。
5. 如还有插入点, 则重复4, 若不再计算, 则键盘输入99999, 程序结束。

### 四、试题

已知两点坐标 (1, 2), (3, 4)

给出  $x=1.25$  得  $Y=2.25$

给出  $x=3.25$  得  $Y=4.25$

### 五、程序清单及运行结果

#### 程序清单

```
7000 PRINT "INPUT KNOWN POINTS ( $X_1, Y_1, X_2, Y_2$ )"
7002 INPUT  $X_1, Y_1, X_2, Y_2$ 
7004 PRINT
7006 PRINT "INPUT X—COORD. OF PT. TO BE INTERPOLATED TO END";
7008 PRINT "END PROGRAM INPUT 99999"
7010 INPUT  $X$ 
7012 PRINT
7014 IF  $X=99999$  THEN GOTO 7024
7016 PRINT "Y=",  $Y_1+(Y_2-Y_1)/(X_2-X_1)*(X-X_1)$ 
```



```

7018 PRINT
7020 PRINT "X",
7022 GOTO 7010
7024 END
运行结果,
RUN
INPUT KNOWN POINTS(X1, Y1, X2, Y2)
? 1? 2? 3? 4
INPUT X—COORD. OF PT. TO BE INTERPOLATED TO END PROGRAM
INPUT 99999
? 1.25
Y=2.25
X? 3.25
Y=4.25
X? 99999
END AT 7024

```

## §2 抛物线插值法

### 一、方法概要

本程序利用三点抛物线插值公式对给定X求出其对应的Y值。

设已知 $f(x)$ 在三个相异的点 $x_0, x_1, x_2$ 的函数值如下表

$$\begin{array}{l} x: x_0 < x_1 < x_2 \\ y: y_0 \quad y_1 \quad y_2 \end{array}$$

我们拟构造一个函数

$$g(x) = Ax^2 + Bx + C$$

使  $g(x_i) = f(x_i) \quad i=0, 1, 2$

则  $g(x)$  就称为  $f(x)$  的抛物线 (或二次) 插值公式。

事实上, 由于所求插值函数的存在及唯一性, 我们可以令

$$g(x) = a + b(x - x_0) + C(x - x_0)(x - x_1)$$

让  $x = x_0$ , 由插值条件  $g(x_0) = f(x_0)$

得  $a = f(x_0)$

再让  $x = x_1$ , 由插值条件  $g(x_1) = f(x_1)$

$$f(x_1) = f(x_0) + b(x_1 - x_0)$$

得: 
$$b = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

让  $x = x_2$ , 由插值条件  $g(x_2) = f(x_2)$

$$f(x_2) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0) + C(x_2 - x_0)(x_2 - x_1)$$

得: 
$$C = \frac{\frac{f(x_2) - f(x_0)}{x_2 - x_0} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_1}$$

这时我们就可以利用所导出的二次插值公式

$$g(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) + \frac{\frac{f(x_2) - f(x_0)}{x_2 - x_0} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_1}(x - x_0)(x - x_1)$$

对任意给出的X, 求出其对应的Y值。

## 二、程序说明

1. 利用数组X(2), Y(2)存放输入的已知结点的坐标值。
2. 此后, 当求取抛物线插值公式时, 利用Y(0)存放a值, 利用Y(1)存放b值, 利用Y(2)存放C值。
3. 本程序使用工作单元: X, Y数组及X<sub>1</sub>, I, K等。

## 三、操作说明

1. 由BASIC解释程序将本程序输入机内。
2. 键盘命令RUN启动本程序执行。电传印出  
"INPUT KNOWN POINTS COORDINATE(X, Y)"
3. 键盘输入已知三点坐标X<sub>0</sub>, Y<sub>0</sub>, X<sub>1</sub>, Y<sub>1</sub>, X<sub>2</sub>, Y<sub>2</sub>。
4. 电传印出  
"INPUT INTERPOLATION POINT X"
5. 键盘输入X值。
6. 电传打印出抛物线插值函数值, 并再次询问新的X值
7. 如还有欲求的插入点, 重复5, 若不再计算, 则键盘输入99999, 程序结束。

## 四、试题

已知三点 (1, 9), (2, 16), (3, 25)

求插值点x=1.5时的抛物线插值的值

计算结果 y=12.25

## 五、程序清单及运行结果

### 程序清单

```
7600 PRINT "INPUT KNOWN POINTS COORDINATE(X,Y)"
7602 DIM X(2),Y(2)
7604 FOR I=0 TO 2
7606   INPUT X(I),Y(I)
7608 NEXT I
7610 LET K=0
7612 PRINT
7614 PRINT "TO END INTERPOLATION,PLEASE INPUT 99999"
7616 PRINT "INPUT INTERPOLATION POINT X"
7618 INPUT X1
7620 PRINT
7622 IF X1=99999 THEN GOTO 7638
7624 IF K><0 THEN GOTO 7632
7626 LET Y(2)=(Y(2)-Y(0))/(X(2)-X(0))
```

```

7628 LET Y(1)=(Y(1)-Y(0))/(X(1)-X(0))
7630 LET Y(2)=(Y(2)-Y(1))/(X(2)-X(1))
7632 PRINT"Y=";Y(0)+(X1-X(0))*Y(1)+(X1-X(0))*(X1-X(1))*Y(2)
7634 LET K=K+1
7636 GOTO 7616
7638 END

```

运行结果

RUN

INPUT KNOWN POINTS COORDINATE(X,Y)

? 1? 9? 2? 16? 3? 25

TO END INTERPOLATION PLEASE INPUT 99999

INPUT INTERPOLATION POINT X

? 1.5

Y=12.25

INPUT INTERPOLATION POINT X

? 4

Y=36

INPUT INTERPOLATION POINT X

? 99999

END AT 7638

### §3 拉格朗日内插法

#### 一、方法概要:

本程序用拉格朗日内插公式对于不等间隔的数据计算出对应于给定的X值的Y值。

$$P(X) = \sum_{k=1}^N \left( \prod_{\substack{i=1 \\ i \neq k}}^N \frac{X - X_i}{X_k - X_i} \right) Y_k$$

P(X)是一多项式, 该多项式的曲线通过全部给定的数据。

#### 二、程序说明

1. 程序使用工作单元X,Y,I,N,Z,K,S,T和数组X(70),Y(70)
2. 计算出结果后, 原始数据被破坏。
3. 点数  $N \leq 70$  (已给定的点数)

#### 三、操作说明

1. 由BASIC解释程序将本程序输入机内。
2. 键盘命令RUN↵, 启动本程序执行。电传机询问已知点数N。
3. 键盘输入N值, 电传询问已知点的X,Y坐标。
4. 键盘输入已知N个点的X,Y坐标, 电传又问内插点的X坐标。
5. 键盘输入内插点的X坐标, 电传机立即打印出对应的Y坐标, 并又询问其它内插点的X坐标。

6. 如继续计算则重复步骤 5, 如不再计算, 则输入 99999, 程序结束。

#### 四、试题

N = 5 各已知点坐标

X	1	2	4	5	6
Y	2	3	5	6	7

内插点坐标  $X=3$ , 计算结果:  $Y=4$ 。

#### 五、程序清单及运行结果

##### 程序清单

```
7200 DIM X(70),Y(70)
7202 PRINT "INPUT NUMBER OF KNOWN POINTS N"
7204 INPUT N
7206 PRINT
7208 PRINT "INPUT KNOWN POINTS(1 POINT TO A LINE)AS FOLLOWS
      X,Y"
7210 FOR I=1 TO N
7212   INPUT X(I),Y(I)
7214 NEXT I
7216 PRINT
7218 PRINT "INPUT X COORDINATE OF POINT TO BE INTERPOLATED"
7220 PRINT "TO END KEY 99999"
7222 PRINT "X";
7224 INPUT Z
7226 PRINT
7228 IF Z=99999 THEN GOTO 7254
7230 LET T=0
7232 FOR K=1 TO N
7234   LET S=1
7236   FOR I=1 TO N
7238     IF I=K THEN GOTO 7242
7240     LET S=S*(Z-X(I))/(X(K)-X(I))
7242   NEXT I
7244   LET T=T+S*Y(K)
7246 NEXT K
7248 PRINT "Y="; T
7250 PRINT
7252 GOTO 7222
7254 END
```

运行结果:

RUN

INPUT NUMBER OF KNOWN POINTS N

? 5

INPUT KNOWN POINTS(1 POINT TO A LINE)AS FOLLOWS X,Y

? 1? 2? 2? 3? 4? 5? 5? 6? 6? 7

INPUT X COORDINATE OF POINT TO BE INTERPOLATED

TO END KEY 99999

X ? 3

Y=4

X ? 8

Y=8.99998

X ? 6

Y=7

X? 99999

END AT 7254

## §4 牛 顿 插 值

### 一、方法概要

本程序是在已知  $n$  个点的坐标值  $(x_i, y_i), i=0, 1, 2, \dots, n-1$  的情况下, 对于  $m$  个已知横坐标为  $x_i, i=1, \dots, m$  的点利用牛顿插值公式求出其对应的  $y_i, i=1, \dots, m$ .

对于给出的一个离散函数  $y_i=f(x_i), i=0, 1, \dots, n-1$ .

$$x_0, x_1, \dots, x_{n-1}$$

$$y_0, y_1, \dots, y_{n-1}$$

欲找出一个不超过  $n-1$  次的多项式  $N(x_i)=f(x_i)$  其中  $i=0, 1, 2, \dots, n-1$

引进记号  $f(x_0, x_1) = \frac{f(x_0) - f(x_1)}{x_0 - x_1}$ , 称之为函数  $f(x)$  在  $x_0, x_1$  的一阶差商.

同样, 引进记号  $f(x_0, x_1, x_2) = \frac{f(x_0, x_1) - f(x_1, x_2)}{x_0 - x_2}$ , 并称之为函数  $f(x)$  在  $x_0, x_1, x_2$  的二阶差商. 一般地说, 有了  $n-1$  阶差商, 可以递归地定义  $n$  阶差商

$$f(x_0, x_1, \dots, x_n) = \frac{f(x_0, x_1, \dots, x_{n-1}) - f(x_1, x_2, \dots, x_n)}{x_0 - x_n}$$

对于已知的  $N$  个点, 由差商的定义可以得知插入点与已知点的关系,

$$\begin{cases} f(x) = f(x_0) + (x - x_0)f(x, x_0) \\ f(x, x_0) = f(x_0, x_1) + (x - x_1)f(x, x_0, x_1) \\ \dots \quad \dots \\ \dots \quad \dots \\ f(x_1, x_0, x_1, \dots, x_{n-2}) = f(x_0, x_1, \dots, x_{n-1}) + (x - x_{n-1})f(x, x_0, \dots, x_{n-1}) \end{cases}$$

将上式中第二式两端乘  $x - x_0$ , 第三式两端乘  $(x - x_0)(x - x_1)$ , 余者类推, 最后相加得

$$\begin{aligned} f(x) = & f(x_0) + (x - x_0)f(x_0, x_1) + \\ & + (x - x_0)(x - x_1)f(x_0, x_1, x_2) + \\ & + \dots + \\ & + (x - x_0)(x - x_1) \dots (x - x_{n-2})f(x_0, x_1, \dots, x_{n-1}) \\ & + R_{n-1}(x) \end{aligned}$$

这里  $R_{n-1}(x) = (x-x_0)(x-x_1)\cdots(x-x_{n-1})f(x, x_0, \cdots, x_{n-1})$

我们记上式前端部分为  $N_{n-1}(x)$ 。

则显然  $N_{n-1}(x)$  即为我们要求的  $n-1$  次多项式。该式亦称为牛顿插值多项式。

## 二、程序说明

1. N 为已知的节点数, M 为欲求的插入点数。
2. 已知节点数据占用工作单元 X(I), Y(I), 在计算完毕后, Y(I) 各单元中所存数据恰好是牛顿插值公式中所要求的各阶差商值。
3. Z(I) 各单元存放所求的插值
4. 本程序使用工作单元: X, Y, Z, C 数组及 N, I, K, M, L, J, S 简单变量。

## 三、操作说明

1. 由 BASIC 解释程序将本程序输入机内。
2. 键盘命令 RUN 启动本程序执行。电传询问已知点数 N。
3. 键盘输入 N, 电传询问已知点的 X, Y 坐标。
4. 键盘输入已知 N 个点的 X, Y 坐标, 电传又问插值点的数目 M。
5. 键盘输入插值点的数目 M, 电传又问插值点的 X 坐标。
6. 键盘输入插值点的 X 坐标, 电传打印出对应的 M 个点的 Y 坐标, 并又询问新的插值点的数目。
7. 如继续计算则重复步骤 5, 如不再计算则输入 M=99999, 程序结束

## 四、试题

N = 4, 各点坐标

X	1	2	3	4
Y	8	27	64	125

M = 3, X = 0, 5, 2.5 计算结果 y = 1, 216, 42.875

## 五、程序清单及运行结果

### 程序清单

```

7400 PRINT "INPUT NUMBER OF KNOWN NODES"
7402 INPUT N
7404 PRINT
7406 PRINT "INPUT KNOWN NODES COORTINATE(X,Y)"
7408 DIM X(N),Y(N)
7410 FOR I=1 TO N
7412 INPUT X(I),Y(I)
7414 NEXT I
7416 LET K=0
7418 PRINT
7420 PRINT "INPUT NUMBER OF INTERPOLATION TO END INTERPOLA-
TION, INPUT 99999"
7422 INPUT M
7424 PRINT

```

```

7426 IF M=99999 THEN GOTO 7482
7428 PRINT "INPUT INTERPOLATION POINTS X"
7430 DIM C(M), Z(M)
7432 FOR I=1 TO M
7434   INPUT C(I)
7436 NEXT I
7438 IF K><0 THEN GOTO 7450
7440 FOR L=2 TO N
7442   FOR I=N TO L STEP -1
7444     LET Y(I)=(Y(I-1)-Y(I))/(X(I-L+1)-X(I))
7446   NEXT I
7448 NEXT L
7450 FOR L=1 TO M
7452   LET Z(L)=0
7454   FOR I=1 TO N
7456     LET S=1
7458     IF I=1 THEN GOTO 7466
7460     FOR J=1 TO I-1
7462       LET S=S*(C(L)-X(J))
7464     NEXT J
7466     LET Z(L)=Z(L)+S*Y(I)
7468   NEXT I
7470 NEXT L
7472 FOR I=1 TO M
7474   PRINT
7476   PRINT "X="; C(I), "Y="; Z(I)
7478 NEXT I
7480 GOTO 7420
7482 END

```

运行结果:

RUN

INPUT NUMBER OF KNOWN NODES

? 4

INPUT KNOWN NODES COORTINATE(X,Y)

? 1? 8? 2? 27? 3? 64? 4? 125

INPUT NUMBER OF INTERPOLATION TO END INTERPOLATION, INPUT

99999

? 3

INPUT INTERPOLATION POINTS X

? 0? 5? 2.5

$X=0$                        $Y=1$   
 $X=5$                        $Y=216$   
 $X=2.5$                     $Y=42.875$

INPUT NUMBER OF INTERPOLATION TO END INTERPOLATION, INPUT  
 99999  
 ? 99999  
 END AT 7482

## §5 三次样条函数插值

### 一、方法概要

样条函数插值除可保证样点的函数值外, 还保证样点上具有一阶和二阶连续导数. 从工程应用的角度来看, 已经相当“光滑”. 因此, 近年来样条函数的应用和发展很快. 在样条函数中, 三次样条函数(也称自然样条)最为常用.

对于给定插值点序列:

$$(x_i, y_i), i=0, 1, 2, \dots, N$$

不妨假定

$$a=x_0 < x_1 < \dots < x_N=b$$

求 $(a, b)$ 上的函数 $S(x)$ 满足下列三个条件:

1.  $S(x_i)=y_i, i=0, 1, \dots, N$
2.  $S(x)$ 在 $(a, b)$ 上有一阶和二阶连续导数, 即在连接点处二阶导数连续.
3. 对 $x \in (x_i, x_{i+1})$ ,  $S(x)$ 为三次多项式, 则 $S(x)$ 即为 $(a, b)$ 上的三次样条插值函数.

令 $m_i$ 表示 $S(x)$ 在 $x_i$ 处的导数值, 即

$$S'(x_i)=m_i, i=0, 1, \dots, N$$

因此, 对 $(a, b)$ 的每一个子区间 $(x_i, x_{i+1})$ 有:

$$\begin{aligned} S(x_i) &= y_i, & S'(x_i) &= m_i \\ S(x_{i+1}) &= y_{i+1}, & S'(x_{i+1}) &= m_{i+1} \end{aligned} \quad \text{成立.}$$

设 $(x_i, x_{i+1})$ 上三次多项式表示为如下形式:

$$P_s(x) = \frac{x-x_i}{x_{i+1}-x_i} y_{i+1} + \frac{x-x_{i+1}}{x_i-x_{i+1}} y_i + (ax+b)(x-x_i)(x-x_{i+1})$$

显然:

$$P_s(x_i)=y_i, \quad P_s(x_{i+1})=y_{i+1}$$

此外:

$$\begin{aligned} P'_s(x) &= \frac{y_{i+1}}{x_{i+1}-x_i} + \frac{y_i}{x_i-x_{i+1}} + a(x-x_i)(x-x_{i+1}) + \\ &\quad + (ax+b)(2x-x_{i+1}-x_i) \end{aligned}$$

则:

$$P'_s(x_i) = \frac{y_{i+1}}{x_{i+1}-x_i} + \frac{y_i}{x_i-x_{i+1}} + (ax_i+b)(x_i-x_{i+1}) = m_i$$



$$\text{而 } ax_i + b = \left( m_i - \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right) / (x_i - x_{i+1})$$

同理:

$$ax_{i+1} + b = \left( m_{i+1} - \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right) / (x_{i+1} - x_i)$$

从上面二式可解出a和b来, 有:

$$ax + b = \frac{(x - x_i)(x - x_{i+1})}{(x - x_i)(x_i - x_{i+1})} \cdot \frac{m_i - \frac{y_{i+1} - y_i}{x_{i+1} - x_i}}{x_i - x_{i+1}} + \frac{(x - x_i)(x - x_{i+1})}{(x - x_{i+1})(x_{i+1} - x_i)} \cdot \frac{m_{i+1} - \frac{y_{i+1} - y_i}{x_{i+1} - x_i}}{x_{i+1} - x_i}$$

将上式代入  $P_s(x)$  中并整理后可得出:

$$\begin{aligned} P_s(x) = & \left[ \frac{3}{h_i^2} (x_{i+1} - x)^2 - \frac{2}{h_i^3} (x_{i+1} - x)^3 \right] \cdot y_i + \\ & + \left[ \frac{3}{h_i^2} (x - x_i)^2 - \frac{2}{h_i^3} (x - x_i)^3 \right] \cdot y_{i+1} + \\ & + h_i \left[ \frac{1}{h_i^2} (x_{i+1} - x)^2 - \frac{1}{h_i^3} (x_{i+1} - x)^3 \right] \cdot m_i - \\ & - h_i \left[ \frac{1}{h_i^2} (x - x_i)^2 - \frac{1}{h_i^3} (x - x_i)^3 \right] \cdot m_{i+1} \end{aligned}$$

其中  $h_i = x_{i+1} - x_i$

因此可见  $(x_i, x_{i+1})$  上的  $P_s(x)$  即为所求的满足给定条件的  $S(x)$ 。以下我们用  $S(x)$  代替所有  $(x_i, x_{i+1})$  上的  $P_s(x)$ ,  $i = 0, 1, \dots, N-1$ 。推导出求各个插值点的一阶导数值  $m_i$  来。

求出  $S(x)$  的二阶导数得:

$$\begin{aligned} S''(x) = & \left[ \frac{6}{h_i^2} - \frac{12}{h_i^3} (x_{i+1} - x) \right] \cdot y_i + \\ & + \left[ \frac{6}{h_i^2} - \frac{12}{h_i^3} (x - x_i) \right] y_{i+1} + \\ & + h \left[ \frac{2}{h_i^2} - \frac{6}{h_i^3} (x_{i+1} - x) \right] \cdot m_i - \\ & - h_i \left[ \frac{2}{h_i^2} - \frac{6}{h_i^3} (x - x_i) \right] m_{i+1} \end{aligned}$$

于是有,

$$S''(x_i) = -\frac{6}{h_i^2} y_i + \frac{6}{h_i^2} y_{i+1} - \frac{4}{h_i} m_i - \frac{2}{h_i} m_{i+1}$$

$$S''(x_{i+1}) = \frac{6}{h_i^2} y_i - \frac{6}{h_i^2} y_{i+1} + \frac{2}{h_i} m_i + \frac{4}{h_i} m_{i+1}$$

此外,  $S(x)$  在子区间  $(x_{i-1}, x_i)$  右端点  $x_i$  的二阶导数值应为:

$$S''(x_i^-) = \frac{6}{h_{i-1}^2} y_{i-1} - \frac{6}{h_{i-1}^2} y_i + \frac{2}{h_{i-1}} m_{i-1} + \frac{4}{h_{i-1}} m_i$$

而  $S(x)$  在子区间  $(x_i, x_{i+1})$  左端点  $x_i$  的二阶导数值应为:

$$S''(x_i^+) = -\frac{6}{h_i^2} y_i + \frac{6}{h_i^2} y_{i+1} - \frac{4}{h_i} m_i - \frac{2}{h_i} m_{i+1}$$

由于要求 $S(x)$ 在连接点 $x_i$ 处二阶导数连续, 应有:

$$S''(x_i^-) = S''(x_i^+)$$

取上述二式相等, 经整理后得:

$$(1-\alpha_i)m_{i-1} + 2m_i + \alpha_i m_{i+1} = 3 \left[ \frac{1-\alpha_i}{h_{i-1}}(y_i - y_{i-1}) + \frac{\alpha_i}{h_i}(y_{i+1} - y_i) \right]$$

$$\text{其中 } \alpha_i = \frac{h_{i-1}}{h_{i-1} + h_i}$$

上式对所有连接点 $x_1, x_2, \dots, x_{N-1}$ 均应成立, 可得到 $N-1$ 个含 $m, i=0, 1, \dots, N$ 的方程, 整理如下:

$$\begin{aligned} (1-\alpha_1)m_0 + 2m_1 + \alpha_1 m_2 &= \beta_1 \\ (1-\alpha_2)m_1 + 2m_2 + \alpha_2 m_3 &= \beta_2 \\ &\vdots \\ (1-\alpha_{N-1})m_{N-2} + 2m_{N-1} + \alpha_{N-1}m_N &= \beta_{N-1} \end{aligned}$$

$$\text{其中 } \beta_i = 3 \left[ \frac{1-\alpha_i}{h_{i-1}}(y_i - y_{i-1}) + \frac{\alpha_i}{h_i}(y_{i+1} - y_i) \right] \quad i=1, 2, \dots, N-1$$

这个方程称为三斜率方程。

若所给的插值问题在 $a, b$ 处满足:

$$S''(x_0) = 0$$

$$S''(x_N) = 0$$

(即在区间 $(a, b)$ 之外, 插值曲线直线延伸)可以由 $S''(x)$ 的表达式得到另外二个方程,

$$\alpha m_0 + m_1 = \frac{3}{h_0}(y_1 - y_0)$$

$$m_{N-1} + 2m_N = \frac{3}{h_{N-1}}(y_N - y_{N-1})$$

与前面所得到的 $N-1$ 个方程联立, 可以求出 $m_0, m_1, \dots, m_N$ , 代入 $S(x)$ 的分段表达式中, 从而求出 $x \in (a, b)$ 的插值函数 $S(x)$ 。

## 二、程序说明

1. 使用数组:  $X, Y, Z, A, B, C, D, H, W$
2. 使用变量:  $N, M, E, E_1, F, F_1, H_1$ 。
3. 程序结束原始信息没有破坏,  $X, Y$ 值保留。 $Z$ 的插值结果在 $W$ 中。

## 三、操作说明

1. 由BASIC解释程序引入本程序, 键盘命令RUN启动本程序, 机器印出:

"INPUT N, M"

询问已知点个数 $N$ 和要插值点个数 $M$ 。

2. 当你用键盘回答 $N$ 和 $M$ 值后, 机器又印出

"INPUT X(I), Y(I)"

询问已知点的 $X$ 值和 $Y$ 值。

3. 用键盘逐点输入 $X$ 值和 $Y$ 值, 直到 $N+1$ 个点送完为止(从0到 $N$ )
4. 机器自动印出解三斜率方程的各系数 $C(I), D(I), A(I), B(I)$ 的值。

5. 机器紧跟着印出已知各点的一阶导数值。

6. 机器印出

“INPUT Z(M)”, 询问要插值的各点  $X_k$  之值

7. 逐点输入所求插值点的  $X$  值, 机器自动印出结果后程序结束。

#### 四、试题

如图11-5-1所示

样条函数为  $y = x^2$

$$N = 4 \quad M = 3$$

$$\begin{cases} x = 0, 1, 2, 3, 4 \\ y = 0, 1, 4, 9, 16 \end{cases}$$

三斜率方程加边界条件:

$$\begin{cases} 2m_0 + m_1 = 3 \\ 0.5m_0 + 2m_1 + 0.5m_2 = 6 \\ 0.5m_1 + 2m_2 + 0.5m_3 = 12 \\ 0.5m_2 + 2m_3 + 0.5m_4 = 18 \\ m_3 + 2m_4 = 21 \end{cases}$$

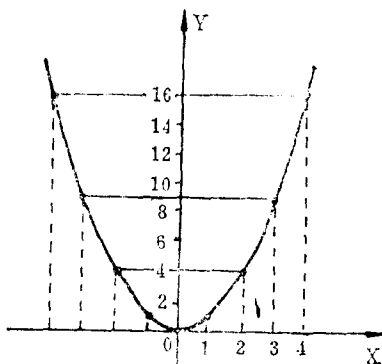


图 11-5-1

得到一阶导数:

$$\begin{cases} m_0 = 0.571428 \\ m_1 = 1.85714 \\ m_2 = 4 \\ m_3 = 6.14286 \\ m_4 = 7.42857 \end{cases}$$

要求的插值点为:

$$\begin{array}{ll} x = 2.5 & y = 6.25 \\ x = 5 & y = 25 \\ x = 3.3 & y = 10.89 \end{array}$$

求出的结果都有一定误差, 由于  $x=5$  时是在给定点之界外, 故误差更大。

#### 五、程序清单及运行结果

程序清单

```
10 PRINT "INPUT N, M"
15 INPUT N, M
20 PRINT
25 DIM X(N+1), Y(N+1), Z(M+1), A(N+1), B(N+1)
26 DIM H(N+1), C(N+1), D(N+1), W(M+1)
35 PRINT "INPUT X(I), Y(I)"
40 FOR I=0 TO N
45   INPUT X(I), Y(I)
50 NEXT I
51 PRINT
55 FOR I=1 TO N
```

```

60     LET H(I-1)=X(I)-X(I-1)
65 NEXT I
70 LET K=N-1
75 FOR I=1 TO K
80     LET A(I)=H(I-1)/(H(I-1)+H(I))
85     LET B(I)=3*((1-A(I))*(Y(I)-Y(I-1))/H(I-1)+A(I)*(Y(I+1)-
        Y(I))/H(I))
90 NEXT I
95 LET A(0)=1
100 LET A(N)=0
105 LET B(0)=3*(Y(1)-Y(0))/H(0)
110 LET B(N)=3*(Y(N)-Y(N-1))/H(N-1)
115 FOR I=0 TO N
120     LET D(I)=2
125 NEXT I
130 FOR I=1 TO N
135     LET C(I)=1-A(I)
136 NEXT I
137 FOR I=0 TO N
138     PRINT C(I), D(I), A(I), B(I)
140 NEXT I
141 LET P=N
145 FOR I=1 TO P
150     IF ABS (D(I))<=.000001 GOTO 288
155     LET A(I-1)=A(I-1)/D(I-1)
160     LET B(I-1)=B(I-1)/D(I-1)
165     LET D(I)=A(I-1)*(-C(I))+D(I)
175     LET B(I)=-C(I)*B(I-1)+B(I)
176 NEXT I
185 LET B(P)=B(P)/D(P)
186 FOR I=1 TO P
190     LET B(P-I)=B(P-I)-A(P-I)*B(P-I+1)
195 NEXT I
196 FOR I=0 TO N
197     PRINT B(I),
198 NEXT I
199 PRINT
200 PRINT "INPUT Z(M)"
201 FOR I=1 TO M
202     INPUT Z(I)

```

```

203 NEXT I
204 FOR I=1 TO M
205   IF Z(I)<X(0) GOTO 220
206   FOR J=1 TO N
210     IF Z(I)<=X(J) GOTO 226
216   NEXT J
218   LET J=N-1
219   GOTO 250
220   LET J=0
221   GOTO 250
226   LET J=J-1
227   GOTO 250
250   LET E=X(J+1)-Z(I)
251   LET E1=E * E
255   LET F=Z(I)-X(J)
256   LET F1=F * F
260   LET H1=H(J) * H(J)
265   LET W(I)=(3 * E1-2 * E1 * E/H(J)) * Y(J)+(3 * F1-2 * F1 * F/
      H(J)) * Y(J+1)
266   LET W(I)=W(I)+(H(J) * E1-E1 * E) * B(J)-(H(J) * F1-F1 * F) *
      B(J+1)
267   LET W(I)=W(I)/H1
270 NEXT I
271 PRINT
272 PRINT "ANS, "
273 PRINT "I", TAB(20); "X", TAB(40); "Y"
275 FOR I=1 TO M
276   PRINT I, TAB(20); Z(I), TAB(40); W(I)
280 NEXT I
282 END
288 PRINT "FAIL"
290 END

```

运行结果:

PUN

INPUT N, M

? 4 ? 3

INPUT X(I), Y(I)

? 0 ? 0 ? 1 ? 1 ? 2 ? 4 ? 3 ? 9 ? 4 ? 16

0	2	1	3
.5	2	.5	6

```

.5          2          .5          12
.5          2          .5          18
1           2           0           21
.571428     1.85714     4           6.14286
7.42857
INPUT Z(M)
? 2.5? 5? 3.3
ANS,
I           X           Y
1           2.5         6.23214
2           5           23
3           3.3         10.947
END AT 290

```

### 参考资料

1. <电子计算机常用算法>

科学出版社

2. <样条函数方法>

李岳生 齐东旭

科学出版社

## 第十二章 数据平滑与滤波

### §1 五点三次平滑

#### 一、基本概念和计算方法:

由于观测数据带有随机误差, 而又要利用这些数据进行其它计算, 就需要根据多次的观测对观测值进行改进, 从而力争消除误差的影响, 提高观测数据的质量, 这就是所谓的数据平滑。

本程序是用五点三次平滑公式, 对等距点上的数据进行平滑。具体公式推导如下:

设  $2n+1$  个等距点  $x_{-n}, x_{-n+1}, \dots, x_{-1}, x_0, x_1, \dots, x_{n-1}, x_n$  上的实验数据为

$y_{-n}, y_{-n+1}, \dots, y_{-1}, y_0, y_1, \dots, y_{n-1}, y_n$ 。若  $h$  为给定点的间距, 作变换  $t = \frac{x-x_0}{h}$ ,

则上述各点将变为  $t_{-n} = -n, t_{-n+1} = -n+1, \dots, t_{-1} = -1, t_0 = 0, t_1 = 1, \dots, t_{n-1} = n-1, t_n = n$

现考虑用  $m$  次多项式

$$y(t) = a_0 + a_1 t + \dots + a_m t^m \quad (12-1-1)$$

来改进所得到的实验数据。为了确定这个多项式的系数  $a_j (j=0, 1, \dots, m)$  使多项式对于所给的数据具有所谓“好的拟合”, 则把所有点  $(t_i, y_i)$  代入多项式 (12-1-1), 就有  $2n+1$  个等式:

$$\begin{cases} a_0 + a_1 t_{-n} + a_2 t_{-n}^2 + \dots + a_m t_{-n}^m - y_{-n} = R_{-n} \\ a_0 + a_1 t_{-n+1} + a_2 t_{-n+1}^2 + \dots + a_m t_{-n+1}^m - y_{-n+1} = R_{-n+1} \\ \dots \\ a_0 + a_1 t_n + a_2 t_n^2 + \dots + a_m t_n^m - y_n = R_n \end{cases}$$

由于拟合多项式曲线不一定通过所有的点  $(t_i, y_i)$ , 所以这些等式不全为 0。最小二乘法提出了一种标志拟合情形好坏的准则。按这个准则来说, 对于  $(2n+1)$  组数据  $(t_i, y_i)$ , 去求系数的最好值, 就是求能使误差  $R_i$  的平方和为最小值的那些  $a_i$  值。

$$\therefore \text{令 } \sum_{i=-n}^n R^2 = \sum_{i=-n}^n \left[ \sum_{j=0}^m a_j t_i^j - y_i \right]^2 = \Phi(a_0, a_1, \dots, a_m) \quad (12-1-2)$$

使  $\Phi(a_0, a_1, \dots, a_m)$  达最小值。

则,  $a_0, a_1, \dots, a_m$  必须满足下列方程组:

$$\frac{\partial \Phi}{\partial a_k} = 2 \sum_{i=-n}^n \left( \sum_{j=0}^m a_j t_i^j - y_i \right) t_i^k = 0 \quad (k=0, 1, 2, \dots, m)$$

$$\text{或 } \sum_{i=-n}^n y_i t_i^k = \sum_{j=0}^m a_j \sum_{i=-n}^n t_i^{k+j} \quad (12-1-3)$$

这个方程组就叫做“正规方程组”

当  $n=2, m=3$  时, 有

$$\begin{cases} 5a_0 + 10a_2 = y_{-2} + y_{-1} + y_0 + y_1 + y_2 \\ 10a_1 + 34a_3 = y_1 - y_{-1} + 2(y_2 - y_{-2}) \\ 10a_0 + 34a_2 = y_1 + y_{-1} + 4(y_2 + y_{-2}) \\ 34a_1 + 130a_3 = y_1 - y_{-1} + 8(y_2 - y_{-2}) \end{cases} \quad (12-1-4)$$

由正规方程 (12-1-4) 解出  $a_0, a_1, a_2, a_3$  再代入 (12-1-1) 并令  $t = 0, \pm 1, \pm 2$  得平滑公式:

$$\bar{y}_{-2} = \frac{1}{70}(69y_{-2} + 4y_{-1} - 6y_0 + 4y_1 - y_2) \quad (12-1-5)$$

$$\bar{y}_{-1} = \frac{1}{35}(2y_{-2} + 27y_{-1} + 12y_0 - 8y_1 + 2y_2) \quad (12-1-6)$$

$$\bar{y}_0 = \frac{1}{35}(-3y_{-2} + 12y_{-1} + 17y_0 + 12y_1 - 3y_2) \quad (12-1-7)$$

$$\bar{y}_1 = \frac{1}{35}(2y_{-2} - 8y_{-1} + 12y_0 + 27y_1 + 2y_2) \quad (12-1-8)$$

$$\bar{y}_2 = \frac{1}{70}(-y_{-2} + 4y_{-1} - 6y_0 + 4y_1 + 69y_2) \quad (12-1-9)$$

其中  $\bar{y}_i$  表示  $y_i$  的改进值。由于  $n=2, 2n+1=5$ , 即取 5 点;  $m=3$ , 即多项式取三次, 故称五点三次平滑公式。

点数很多时, 为对称起见, 除两端点附近外, 都用公式 (12-1-7), 在两端点则分别用公式 (12-1-5), (12-1-6) 和 (12-1-8), (12-1-9) 进行平滑。这就相当于在每个子区间上用不同的三次最小二乘多项式进行平滑。

从以上推导来看, 具体平滑公式中只用  $y_i$  值, 并不用间隔值  $h$  和  $x_i$  之值。

## 二、程序说明

1. 给定点的个数  $n \geq 5$
2. 本程序中使用数组 A 和 B, 输入  $y$  值分别存放在 A、B 两数组中。程序结束时 A 数组中存放结果值, B 数组中保持信息不变。

## 三、使用说明

1. 由 BASIC 解释程序引入本程序, 用 RUN 启动本程序执行, 机器印出: "INPUT NUMBER N", 询问实验数据的点数。
2. 由键盘回答点数 N 之值, 机器又自动印出: "INPUT VALUE OF FUNC. Y", 要求输入各点相应的实验数据值, 从  $x_{-2}$  到  $x_n$  顺序输入各点对应的  $y$  值。

## 四、例题

已知实验数据:

$x_i$	0	1	2	3	4	5	6	7	8
$y_i$	54	145	227	359	401	342	259	112	65

经平滑后的数据:

$x_i$	0	1	2	3	4	5	6	7	8
$y_i$	56.8	133.6	244.1	347.9	393.5	352	241.5	123.7	62.1

## 五、程序清单及运行结果

程序清单:



```

8000 REM SMOTH
8002 PRINT "INPUT NUMBER N"
8004 INPUT N
8006 PRINT
8008 PRINT "INPUT VALUE OF FUNC. Y"
8010 PRINT
8012 DIM A(N)
8014 FOR I=1 TO N
8016   INPUT A(I)
8018 NEXT I
8020 PRINT
8022 FOR I=1 TO N
8024   LET B(I)=A(I)
8026 NEXT I
8028   LET A(1)=(69*B(1)+4*(B(2)+B(4))-6*B(3)-B(5))/70
8030   LET A(2)=(2*(B(1)+B(5))+27*B(2)+12*B(3)-8*B(4))/35
8032   FOR I=3 TO N-2
8034     LET A(I)=(-3*(B(I-2)+B(I+2))+12*(B(I-1)+B(I+1))
               +17*B(I))/35
8036   NEXT I
8038   LET A(N-1)=(2*(B(N-4)+B(N))-8*B(N-3)+12*B(N-2)
               -27*B(N-1))/35
8040   LET A(N)=(-B(N-4)+4*(B(N-3)+B(N-1))-6*B(N-2)+
               69*B(N))/70
8042 PRINT "NO. ", "OLD", "NEW"
8044 FOR I=1 TO N
8046   PRINT I-1, B(I), A(I)
8048 NEXT I
8050 END

```

运行结果:

RUN

INPUT NUMBER N

? 9

INPUT VALUE OF FUNC. Y

? 54 ? 145 ? 227 ? 359 ? 401 ? 342 ? 259 ? 112 ? 65

NO.	OLD	NEW
0	54	56.8428
1	145	133.629
2	227	244.057
3	359	347.943

4	401	393.457
5	342	352.029
6	259	241.514
7	112	123.657
8	65	62.0857

END AT 8050

参考资料:

《电子计算机常用算法》 科学出版社 1976

## §2 $\alpha$ - $\beta$ - $\gamma$ 滤波器

### 一、方法概述

Kalman滤波方法的缺点之一,是需要进行大量的实时计算,通过解矩阵Riccati方程求得最优增益矩阵,这给实际应用带来了一定的困难。为了避免这一点,人们常常希望使用常增益的Kalman滤波技术,其中最简单的情形就是 $\alpha$ - $\beta$ - $\gamma$ 滤波器。

使用 $\alpha$ - $\beta$ - $\gamma$ 滤波器所要解决的问题可以用数学语言描述如下:

假设一个过程的量测数据为 $S^*(t)$ ,它是有用讯号 $S(t)$ 和量测噪声 $V(t)$ 的混合,即

$$S^*(t) = S(t) + V(t) \quad (12-2-1)$$

并且设 $V(t)$ 是一个均值为零的白噪声过程

$$\text{即有: } E\{V(t)\} = 0$$

$$E\{V(t)V(\tau)\} = \gamma\delta(t-\tau) \quad (12-2-2)$$

问题是怎样根据量测数据 $S^*(t)$ 把有用讯号 $S(t)$ 估计出来。

显然这类问题在实际应用中是大量存在的。如果我们用一次直线去近似 $S(t)$ ,使用的就是 $\alpha$ - $\beta$ 滤波器;若用二次曲线近似 $S(t)$ ,则使用的就是 $\alpha$ - $\beta$ - $\gamma$ 滤波器。

使用 $\alpha$ - $\beta$ - $\gamma$ 滤波器运算公式如下:

滤波估值方程:

$$\left\{ \begin{array}{l} \hat{x}_{n+1} = \hat{x}_{n+1/n} + \alpha(x_{n+1} - \hat{x}_{n+1/n}) \\ \hat{\dot{x}}_{n+1} = \hat{\dot{x}}_{n+1/n} + \frac{\beta}{T}(x_{n+1} - \hat{x}_{n+1/n}) \\ \hat{\ddot{x}}_{n+1} = \hat{\ddot{x}}_{n+1/n} + \frac{2\gamma}{T^2}(x_{n+1} - \hat{x}_{n+1/n}) \end{array} \right.$$

一步预测估值方程:

$$\hat{x}_{n+1/n} = \hat{x}_n + \hat{\dot{x}}_n T + \hat{\ddot{x}}_n \frac{T^2}{2}$$

$$\hat{\dot{x}}_{n+1/n} = \hat{\dot{x}}_n + \hat{\ddot{x}}_n T$$

$$\hat{\ddot{x}}_{n+1/n} = \hat{\ddot{x}}_n \quad (12-2-3)$$

其中:  $x_{n+1}$ : 本时刻的测量采样值

$\hat{x}_{n+1}$ : 本时刻的滤波估值

$\hat{x}_{n+1/n}$ : 上时刻对本时刻的一步预测估值

$T$ : 采样周期

$\alpha, \beta, \gamma$ : 反映滤波器的结构的常系数

$\alpha, \beta, \gamma$ 数值的选取直接影响到滤波器的性能, 因此选取  $\alpha, \beta, \gamma$  是滤波器使用者进行系统设计要完成的主要任务。下面我们比较简要地说明一种  $\alpha, \beta, \gamma$  参数选取的办法。

我们可以推导出位置、速度、加速度估值对应输入测量值的等效传递函数为:

$$G_1(z) = \frac{\alpha z^2 + (-2\alpha + \beta + \gamma)z + (\alpha - \beta + \gamma)z}{z^3 - (3 - \alpha - \beta - \gamma)z^2 + (3 - 2\alpha - \beta + \gamma)z - (1 - \alpha)}$$

$$G_2(z) = \frac{\left(\frac{\beta}{T}z\right)(z-1)\left(z + \frac{2\gamma - \beta}{\beta}\right)}{z^3 - (3 - \alpha - \beta - \gamma)z^2 + (3 - 2\alpha - \beta + \gamma)z - (1 - \alpha)}$$

$$G_3(z) = \frac{\left(\frac{2\gamma}{T^2}z\right)(z-1)(z-1)}{z^3 - (3 - \alpha - \beta - \gamma)z^2 + (3 - 2\alpha - \beta + \gamma)z - (1 - \alpha)}$$

而一步预测估值(下标用 P 记)对应测量值的等效传递函数分别为:

$$G_{1p}(z) = G_1 + T G_2 + \frac{T^2}{2} G_3 \quad (12-2-4)$$

$$G_{2p}(z) = G_2 + T G_3$$

$$G_{3p}(z) = G_3$$

可以把预测估值对应测量值的等效传递函数, 写成下列规一化形式, 即:

$$G_p(z) = \frac{a_3 z^3 + a_2 z^2 + a_1 z}{z^3 + b_2 z^2 + b_1 z + b_0} \quad (12-2-5)$$

其中各系数值见下表 12-1

表 12-1

系 数	$a_3$	$a_2$	$a_1$	$b_2$	$b_1$	$b_0$
予 测 位 置	$\alpha + \beta + \gamma$	$\gamma - 2\alpha - \beta$	$\alpha$	$\alpha + \beta + \gamma - 3$	$3 - 2\alpha - \beta + \gamma$	$\alpha - 1$
予 测 速 度	$\frac{\beta + 2\gamma}{T}$	$\frac{-2\beta - 2\gamma}{T}$	$\frac{\beta}{T}$	$\alpha + \beta + \gamma - 3$	$3 - 2\alpha - \beta + \gamma$	$\alpha - 1$
予 测 加 速 度	$\frac{2\gamma}{T^2}$	$-\frac{4\gamma}{T^2}$	$\frac{2\gamma}{T^2}$	$\alpha + \beta + \gamma - 3$	$3 - 2\alpha - \beta + \gamma$	$\alpha - 1$

从以上可以看出所有传递函数的特征方程为:

$$C(z) = z^3 - (3 - \alpha - \beta - \gamma)z^2 + (3 - 2\alpha - \beta + \gamma)z - (1 - \alpha) = 0$$

由稳定性分析可知, 系统的稳定条件为:

$$\alpha > 0, \beta > 0, \gamma > 0, 2\alpha + \beta \leq 4, 2\alpha > \beta, \alpha(\beta + \gamma) > 2\gamma$$

为具体选取  $\alpha, \beta, \gamma$  参数方便, 我们又可以把  $\alpha - \beta - \gamma$  系统的等效传递函数, 写成如下表达式

$$G(z) = \frac{(\cdot)}{(z-d)(z^2 - 2ze^{-\zeta\omega_0 T} \cos \omega_d T + e^{-2\zeta\omega_0 T})} \quad (12-2-6)$$

式中  $d$  为  $0 \sim 1$  之间的正实根,

$\zeta$  为阻尼系数

$\omega_0$  自然谐振频率

$\omega_d$  阻尼振荡频率。

且  $\omega_d = \omega_0 \sqrt{1 - \zeta^2}$

这些参数与  $\alpha$ 、 $\beta$ 、 $\gamma$  参数的关系如下:

$$\begin{cases} \alpha = 1 - de^{-2\zeta\omega_0 T} \\ \beta = \frac{(3-d)}{2} - (1+d)e^{-\zeta\omega_0 T} \cos(\omega_d T) - \frac{1}{2}(1-3d)e^{-2\zeta\omega_0 T} \\ 2\gamma = (1-d)(1 + e^{-2\zeta\omega_0 T} - 2e^{-\zeta\omega_0 T} \cos(\omega_d T)) \end{cases} \quad (12-2-7)$$

显然,  $d$  不应该选成负值。若  $d = 1$ ,  $\gamma$  值为零值, 则  $\alpha$ - $\beta$ - $\gamma$  系统变成  $\alpha$ - $\beta$  系统; 若  $d = 0$ ,  $\alpha = 1$ ,  $G_X = 1$ , 系统成为一个全通滤波器, 没有平滑作用。 $d$  不应选成大于 1, 否则将出现不稳定。

这样, 对  $\alpha$ ,  $\beta$ ,  $\gamma$  选取就变成对  $\omega_0$ ,  $\zeta$ ,  $d$  的选取, 这在控制系统设计中已是容易的事情了。(注: 以上对  $\alpha$ ,  $\beta$ ,  $\gamma$  选取的方法仅供使用者参考, 本节并不包括选取  $\alpha$ ,  $\beta$ ,  $\gamma$  的程序。)

## 二、程序说明

从 8030 到 8042 语句是  $\alpha$ - $\beta$ - $\gamma$  滤波器的工作子程序; 而 8000 到 8028 语句, 是假想的一个主程序。主程序每隔一个采样间隔时间, 采样一个目标信号, 即取自定义函数  $FNA(X)$  之值; 然后调用  $\alpha$ - $\beta$ - $\gamma$  滤波器工作子程序对采样值进行滤波估计。在  $T > 1.5$  秒 (1.5 秒认为是滤波器的过渡过程时间) 印出目标信号的位置、速度和加速度之值, 以及经过滤波器后估计出来的位置、速度和加速度的估计值和一步预测值。

本程序使用了一维数组  $D(10)$ , 使用简单变量有  $T, K$ 。 $D(10)$  中各元素代表的数值如下

$D(1)$ —— $\alpha$ 值	$D(6)$ ——速度估计值
$D(2)$ —— $\beta$ 值	$D(7)$ ——加速度估计值
$D(3)$ —— $\gamma$ 值	$D(8)$ ——位置一步预测值
$D(4)$ ——采样间隔值	$D(9)$ ——速度一步预测值
$D(5)$ ——位置估计值	$D(10)$ ——加速度一步预测值

## 三、程序使用操作

1. 用 BASIC 解释程序将本程序引入, 由键盘命令 `RUN` 启动运行, 机器自动印出:  
" INPUT ALPHA, BETA, GAMMA AND DELTA T" 询问  $\alpha$ ,  $\beta$ ,  $\gamma$  值及采样间隔值  $T$ 。
2. 用键盘回答以上各值后, 机器将从  $T > 1.5$  秒开始, 每隔一个采样周期  $T$  印出以下各值, 按印出先后的顺序为:  
时间  $T$ , 目标的位置值、速度值、加速度值  
位置估值、速度估值、加速度估值  
位置一步预测值、速度一步预测值、加速度一步预测值

## 四、例题

$$\text{目标信号 } S(t) = 5 - 2t + 3t^2$$

## 五、程序清单和运行结果

### 程序清单

```

8000 DEF FNA(X)=5-2*X+3*X*X
8002 DIM D(10)
8004 PRINT "INPUT ALPHA, BLTA, GAMMA AND DELTA T"
8006 INPUT D(1), D(2), D(3), D(4)
8008 PRINT
8010 FOR K=1 TO 100
8012 GOSUB 8030
8014 IF T<1.5 THEN GOTO 8024
8016 PRINT "T=", T
8018 PRINT FNA(T), -2+6*T, 6
8020 PRINT D(5), D(6), D(7)
8022 PRINT D(8), D(9), D(10)
8024 LET T=T+D(4)
8026 NEXT K
8028 END
8030 LET D(8)=D(5)+D(4)*D(6)+D(4)*D(4)/2*D(7)
8032 LET D(9)=D(6)+D(4)*D(7)
8034 LET D(10)=D(7)
8036 LET D(5)=D(8)+D(1)*(FNA(T)-D(8))
8038 LET D(6)=D(9)+D(2)/D(4)*(FNA(T)-D(8))
8040 LET D(7)=D(10)+2*D(3)/D(4)/D(4)*(FNA(T)-D(8))
8042 RETURN

```

运行结果如下:

RUN (注: 这里结果从  $T = 0$  印出, 目的使大家看到滤波器的过渡过程)

INPUT ALPHA, BLTA, GAMMA AND DELTA T

? .488? .108? .004? .04

T = 0

5	-2	6
2.44	13.5	25
0	0	0

T = .04

4.9248	-1.76	6
3.9393	19.697	34.624
3	14.5	25

T = .08

4.8592	-1.52	6
4.80579	21.3636	35.1456

4.75488	21.0819	34.624
T = .12		
4.8032	--1.28	6
5.25644	20.3792	30.7194
5.68844	22.7694	35.1456
T = .16		
4.7568	--1.04	6
5.44257	17.9917	24.0224
6.09619	21.608	30.7194
T = .2		
4.72	--.8	6
5.46826	15.0066	16.7152
6.18145	18.9525	24.0224
T = .24		
4.6928	--.56	6
5.40402	11.9247	9.76968
6.0819	15.6752	16.7152
T = .28		
4.6752	--.320001	6
5.29657	9.03869	3.70158
5.88882	12.3155	9.76968
T = .32		
4.6672	--8.00009 E-2	6
5.17607	6.50327	--1.26782
5.66108	9.18675	3.70158
T = .36		
4.6688	.159999	6
5.06119	4.38333	--5.09973
5.43518	6.45256	--1.26782
T = .4		
4.68	.399999	6
4.96285	2.68775	--7.86193
5.23244	4.17934	--5.09973
T = .44		
4.7008	.639998	6
4.88679	1.39245	--9.67827
5.06407	2.37327	--7.86193
T = .48		
4.7312	.879998	6
4.83541	.455735	--10.696

4.93475	1.00532	-9.67827
T = .52		
4.7712	1.12	6
4.80903	- .1716	-11.0654
4.84509	2.78941 E-2	-10.696
T = 1.52		
8.89119	7.11999	6
8.8749	6.86412	5.11143
8.85937	6.77821	4.95235
T = 1.56		
9.18079	7.35999	6
9.16684	7.14212	5.24762
9.15355	7.06857	5.11143
T = 1.6		
9.47999	7.59999	6
9.46808	7.41483	5.36394
9.45672	7.35202	5.24762
T = 1.64		
9.78879	7.83999	6
9.77863	7.68292	5.46308
9.76896	7.62939	5.36394
T = 1.68		
10.1072	8.07999	6
10.0985	7.94699	5.54741
10.0903	7.90145	5.46308
T = 1.72		
10.4352	8.31999	6
10.4270	8.20755	5.61901
10.4209	8.16888	5.54741
T = 1.76		
10.7728	8.55999	6
10.7666	8.46508	5.67971
10.7606	8.43231	5.61901
T = 1.8		
11.12	8.79999	6
11.1147	8.72	5.73106
11.1097	8.69227	5.67971
T = 1.84		
11.4768	9.03999	6
11.4723	8.97266	5.77444

11.4681	8.94924	5.73106
T = 1.88		
11.8432	9.27999	6
11.8394	9.2234	5.81103
11.8359	9.20364	5.77444
T = 1.92		
12.2192	9.51999	6
12.216	9.47248	5.84185
12.213	9.45584	5.81103
T = 1.96		
12.6048	9.75999	6
12.6021	9.72015	5.86777
12.5996	9.70616	5.84185
T = 2		
13	9.99999	6
12.9977	9.96662	5.88955
12.9956	9.95486	5.86777
T = 2.04		
13.4048	10.24	6
13.4029	10.2121	5.90782
13.4011	10.2022	5.88955

END AT 8028

六、为考验 $\alpha$ - $\beta$ - $\gamma$ 滤波器抑制噪声干扰的能力，特别编制了一个使用 $\alpha$ - $\beta$ - $\gamma$ 滤波器的新的主程序（语句标号从8000到8044）。在这个主程序中，在有用的目标信号上，人为地送加上一个均值为零、标准差为0.5的正态分布白噪声 $Y(t)$ ，这是由8046到8056语句组成的子程序产生的。（有关正态白噪声产生技术问题见本书第六章有关内容）。在主程序中又加上了检验滤波器滤除干扰程度的检验部分，即语句8020~8026几句。这几个语句的功能就是从 $T > 3$ 秒开始，将每一采样间隔的各个估计值与其对应的真值的差求绝对值的和，然后再取平均值：

$$S = \frac{\sum_{i=1}^N |\hat{x}_i - x_i|}{N} \quad (12-2-8)$$

我们使用这个程序，选了几组不同的 $\alpha$ 、 $\beta$ 、 $\gamma$ 值运行，将其所得结果进行比较。所选用的 $\alpha$ 、 $\beta$ 、 $\gamma$ 各组值如表12-2所示。

从运行的结果来看，第5组参数抑制噪声最好。



表 12-2

序 号	$\xi$	$\omega_0$	d	$\alpha$	$\beta$	$\gamma$
1	0.7	5	0.6	0.54653	0.125505	0.00695453
2	0.7	3	0.6	0.492788	0.0724503	0.00264792
3	0.7	2	0.6	0.463574	0.0472239	0.00121031
4				0.488	0.108	0.004
5				0.271	0.0285	0.0005

程序及运行结果如下:

程序清单

```

8000 DEF FNA(X)=5-2*X+3*X*X+Y
8002 DIM D(10), S(6)
8004 PRINT "INPUT ALPHA, BETA, GAMMA AND DELTA T"
8006 INPUT D(1), D(2), D(3), D(4)
8008 PRINT
8010 FOR K=1 TO 30
8012   FOR L=1 TO 5
8014     LET T=T+D(4)
8016     GOSUB 8046
8018     GOSUB 8058
8020     IF T<3 THEN GOTO 8028
8022     LET S(1)=S(1)+ABS(D(5)-FNA(T)+Y)
8024     LET S(2)=S(2)+ABS(D(6)+2-6*T)
8026     LET S(3)=S(3)+ABS(D(7)-6)
8028   NEXT L
8030   IF T<3 THEN GOTO 8040
8032   PRINT " T=", T
8034   PRINT FNA(T)-Y, -2+6*T, 6
8036   PRINT D(5), D(6), D(7)
8038   PRINT D(8), D(9), D(10)
8040 NEXT K
8042 PRINT "S(1)=", S(1)/75, "S(2)=", S(2)/75, "S(3)=", S(3)/75
8044 END
8046 LET S8=0
8048 FOR l=1 TO 12

```

```

8050 LET S8=S8+RND(0)
8052 NEXT J
8054 LET Y=.5*(S8-6)
8056 RETURN
8058 LET D(8)=D(5)+D(4)*D(6)+D(4)*D(4)/2*D(7)
8060 LET D(9)=D(6)+D(4)*D(7)
8062 LET D(10)=D(7)
8064 LET D(5)=D(8)+D(1)*(FNA(T)-D(8))
8066 LET D(6)=D(9)+D(2)/D(4)*(FNA(T)-D(8))
8068 LET D(7)=D(10)+2*D(3)/D(4)/D(4)*(FNA(T)-D(8))
8070 RETURN

```

第一组参数运行结果

RUN

INPUT ALPHA, BETA, GAMMA AND DELTA T

? .54653 ? .125505 ? 6.95453E-3 ? .04

T = 3.2

29.3199	17.2	6
29.5135	18.3226	9.11326
29.4327	17.8587	7.82798

T = 3.4

32.8799	18.4	6
32.9705	19.1952	8.58698
32.8542	18.5274	6.73649

T = 3.6

36.6799	19.6	6
36.4606	18.9909	5.14347
36.1817	17.3898	.707415

T = 3.8

40.7199	20.8	6
40.4986	20.2548	4.66507
40.8515	22.2822	10.2821

T = 4

44.9999	22	6
45.403	24.7122	13.0523
44.8556	21.5696	4.34539

T = 4.2

49.5199	23.2	6
49.5773	23.4517	6.31248
49.1916	21.2369	.176089

T = 4.4

54.2799	24.4	6
53.8185	22.1381	.472554
54.1105	23.8148	5.11799
T = 4.6		
59.2799	25.6	6
59.6668	26.6653	6.23991
60.2978	30.2878	16.2766
T = 4.8		
64.5199	26.8	6
64.3493	25.1709	1.61265
64.3499	25.1745	1.6226
T = 5		
69.9998	28	6
69.8883	28.1222	8.51043
69.8226	27.7452	7.46584
T = 5.2		
75.7198	29.2	6
75.8518	30.6689	10.8245
75.4621	28.4311	4.62427
T = 5.39999		
81.6798	30.4	6
81.676	30.7765	6.89598
81.5312	29.9454	4.59336
T = 5.59999		
87.8798	31.6	6
87.6392	31.0383	4.95496
87.7402	31.6178	6.56052
T = 5.79999		
94.3198	32.8	6
94.3326	33.1526	5.99769
94.4886	34.0482	8.47899
T = 5.99999		
101	34	6
101.107	34.6858	7.4271
100.646	32.0404	.097683
S(1) = .217312	S(2) = 1.34858	S(3) = 3.79659
END AT 8044		

第二组参数运行结果

INPUT ALPHA, BETA, GAMMA AND DELTA T

‡ .492788 ‡ 7.24503E-2 ‡ 2.64792E-3 ‡ .04

T = 3.2		
29.3199	17.2	6
29.3097	16.9171	4.88458
29.5537	17.8139	6.52347
T = 3.4		
32.8799	18.4	6
32.9128	18.2477	5.38092
32.9817	18.5009	5.84357
T = 3.6		
36.6799	19.6	6
36.7185	19.5291	5.79774
36.6746	19.3676	5.50249
T = 3.8		
40.7199	20.8	6
40.7015	21.0214	7.069
40.3396	19.6912	4.63817
T = 4		
44.9999	22	6
44.7185	21.3242	5.24536
44.423	20.2379	3.26034
T = 4.2		
49.5199	23.2	6
49.5327	23.7069	7.23622
49.7007	24.3244	8.3646
T = 4.4		
54.2799	24.4	6
53.9471	23.5006	4.35483
54.0557	23.8995	5.08382
T = 4.6		
59.2799	25.6	6
59.3456	26.022	6.50899
59.4876	26.544	7.46285
T = 4.8		
64.5199	26.8	6
64.1628	25.353	2.83071
64.6106	26.9989	5.83843
T = 5		
69.9998	28	6
69.9803	27.8202	5.40535
69.8711	27.4188	4.67177

T = 5.2

75.7198	29.2	6
75.4628	28.2419	4.2659
75.7938	29.4586	6.48939

T = 5.39999

81.6798	30.4	6
81.7383	30.7248	6.75762
81.4455	29.6484	4.7907

T = 5.59999

87.8798	31.6	6
88.0871	32.4534	7.58348
88.0784	32.4215	7.52524

T = 5.79999

94.3198	32.8	6
93.529	29.6328	- .189807
94.4892	33.1623	6.25999

T = 5.99999

101	34	6
100.854	33.2571	4.33367
101.564	35.8654	9.10004

S(1) = .197649      S(2) = .764701      S(3) = 1.4591

END AT 8044

第三组参数运行结果

RUN

INPUT ALPHA, BETA, GAMMA AND DELTA T

? .463574 ? 4.72239E-2 ? 1.21031E-3 ? .04

T = 3.2

29.3199	17.2	6
29.0456	16.8775	5.91496
29.3474	17.6462	6.90004

T = 3.4

32.8799	18.4	6
32.7105	18.5187	6.53274
32.5589	18.1327	6.03805

T = 3.6

36.6799	19.6	6
36.78	20.3855	7.20283
36.837	20.5308	7.38903

T = 3.8

40.7199	20.8	6
---------	------	---

40.7436	21.2283	6.50692
40.5821	20.817	5.97983
T = 4		
44.9999	22	6
45.145	22.5618	6.51031
45.0236	22.2527	6.11423
T = 4.2		
49.5199	23.2	6
49.5513	23.2761	5.77676
49.5708	23.3257	5.84024
T = 4.4		
54.2799	24.4	6
53.7369	22.9609	3.91498
54.0189	23.6791	4.83526
T = 4.6		
59.2799	25.6	6
59.6305	26.4215	6.86169
59.5869	26.3106	6.71962
T = 4.8		
64.5199	26.8	6
64.0592	25.6056	4.43813
64.0329	25.5388	4.35246
T = 5		
69.9998	28	6
70.2166	28.569	6.7298
70.0697	28.1949	6.25031
T = 5.2		
75.7198	29.2	6
75.607	28.9736	5.75821
75.8531	29.6003	6.56126
T = 5.39999		
81.6798	30.4	6
81.479	29.9854	5.52851
81.7666	30.7178	6.4671
T = 5.59999		
87.8798	31.6	6
88.3154	32.9129	7.81075
88.0619	32.2674	6.98359
T = 5.79999		
94.3198	32.8	6

94.3757	33.0223	6.25238
94.3033	32.8379	6.01606
T = 5.99999		
101	34	6
101.217	34.5029	6.47817
101.449	35.0932	7.23461
S(1) = .205424	S(2) = .544068	S(3) = .70718
END AT 8044		

#### 第四组参数运行结果

RUN

INPUT ALPHA, BETA, GAMMA AND DELTA T

? .488 ? .108 ? .004 ? .04

T = 3.2

29.3199	17.2	6
28.4984	12.7648	-2.59521
29.0633	15.8902	3.19255

T = 3.4

32.8799	18.4	6
33.0882	20.4322	9.98492
32.8879	19.3236	7.93196

T = 3.6

36.6799	19.6	6
36.5044	18.5641	3.58599
36.6709	19.4853	5.29192

T = 3.8

40.7199	20.8	6
40.9807	21.824	7.34613
40.7246	20.4074	4.72281

T = 4

44.9999	22	6
44.9626	21.657	5.25095
45.1105	22.4753	6.76622

T = 4.2

49.5199	23.2	6
49.2865	21.7915	3.35413
49.3176	21.9635	3.67269

T = 4.4

54.2799	24.4	6
54.0448	23.3822	4.33836
54.2412	24.4685	6.35007

T = 4.6		
59.2799	25.6	6
59.2133	26.2421	7.76319
59.119	25.7206	6.79742
T = 4.8		
64.5199	26.8	6
64.5901	27.6391	7.51664
64.2718	25.878	4.25531
T = 5		
69.9998	28	6
69.886	27.5048	4.77219
70.326	29.9395	9.28094
T = 5.2		
75.7198	29.2	6
75.6319	28.5771	4.50265
75.6976	28.9407	5.17602
T = 5.39999		
81.6798	30.4	6
81.4939	30.2556	6.34362
80.9256	27.1117	.521477
T = 5.59999		
87.8798	31.6	6
88.7181	36.1824	14.0854
88.2057	33.3474	8.83545
T = 5.79999		
94.3198	32.8	6
94.6573	33.4916	6.39305
94.5121	32.6881	4.90509
T = 5.99999		
101	34	6
100.721	32.5559	3.82889
100.453	31.074	1.08444
S(1) = .209669	S(2) = 1.22647	S(3) = 2.36839
END AT 8044		

第五组参数运行结果

RUN

INPUT ALPHA, BETA, GAMMA AND DELTA T

? .271 ? .0285 ? .0005 ? .04

T = 3.2

29.3199	17.2	6
---------	------	---



29.0606	16.8115	5.61268
29.1913	17.1552	5.91415
<b>T = 3.4</b>		
32.8799	18.4	6
32.8686	18.6371	6.16195
32.7064	18.2107	5.78794
<b>T = 3.6</b>		
36.6799	19.6	6
36.3781	19.0953	5.53177
36.5042	19.4267	5.82241
<b>T = 3.8</b>		
40.7199	20.8	6
41.0299	21.8387	6.86361
40.7949	21.2209	6.32175
<b>T = 4</b>		
44.9999	22	6
45.67	23.4906	7.07902
45.5759	23.2432	6.86203
<b>T = 4.2</b>		
49.5199	23.2	6
49.6856	23.0726	5.606
49.7276	23.1829	5.70277
<b>T = 4.4</b>		
54.2799	24.4	6
54.2859	23.8723	5.32912
54.2581	23.7991	5.2649
<b>T = 4.6</b>		
59.2799	25.6	6
59.2506	25.1993	5.57497
59.144	24.9191	5.32916
<b>T = 4.8</b>		
64.5199	26.8	6
64.327	25.9832	5.23217
64.5036	26.4473	5.63924
<b>T = 5</b>		
69.9998	28	6
70.0291	28.0279	6.08532
69.8334	27.5133	5.63396
<b>T = 5.2</b>		
75.7198	29.2	6

75.2731	28.064	5.08699
75.4128	28.4312	5.40905
T = 5.39999		
81.6798	30.4	6
81.4986	30.2615	6.05518
81.4506	30.1353	5.9445
T = 5.59999		
87.8798	31.6	6
87.6827	31.369	5.91342
87.752	31.5512	6.0732
T = 5.79999		
94.3198	32.8	6
94.3716	33.3415	6.60975
94.0437	32.4796	5.85365
T = 5.99999		
101	34	6
100.864	33.9224	5.9891
100.943	34.1321	6.17311
S(1) = .18786	S(2) = .451368	S(3) = .407561
END AT 8044		

### §3 离散随机线性系统的Kalman滤波器

#### 一、方法概述

许多实际的物理量测系统都是在时间的离散点上进行采样的，因此，由量测系统获得的采样数据都不是连续的。那么使用这样的量测数据进行滤波时，往往使用连续系统的Kalman滤波方法是不行的，尤其是采样频率低的情况下更是这样。这时就要使用离散随机线性系统的Kalman滤波方法，这种方法又叫做极小方差线性递推滤波方法。

在这里我们不做严格地理论推导，而从使用者的角度作一个直观的介绍。

离散随机线性系统是由差分方程组描述的，即：

$$X_k = \Phi_{k,k-1} X_{k-1} + W_{k-1} \quad (12-3-1)$$

$$Y_k = H_k X_k + V_k \quad (12-3-2)$$

$$k=1, 2, \dots$$

在(12-3-1)式中， $\Phi_{k,k-1}$ 是一个 $n \times n$ 阶矩阵，称为系统的状态转移矩阵(或状态跃迁阵)，它反映了系统从第 $k-1$ 个采样时刻的状态到第 $k$ 个采样时刻的状态的变换； $X_{k-1}$ 表示系统在第 $k-1$ 时刻的状态； $W_{k-1}$ 是 $n$ 维矢量，它表示在第 $k-1$ 时刻作用于系统的随机干扰。 $W_{k-1}$ 也是一种输入，但它是一种随机干扰输入。如果要考虑控制输入，则在(12-3-1)式的右边还应添入一个确定性的控制输入项 $B_{k-1}U_{k-1}$ ，这里暂时不考虑。在实际的问题中， $W_{k-1}$ 能表现为作用于系统的随机干扰，也能表现为由于模型近似而带来的模型误差，叫做模型噪声。为简单起见，假设随机序列 $\{W_k\}$ 为高斯白噪声序列，具有已知的期望和协方差阵：

$$E\{W_k\}=0$$

$$E\{W_k W_k^*\}=Q_k \delta_{ki}$$

其中 $Q_k$ 是一个 $n \times n$ 阶的非负定矩阵,称为模型噪声的协方差矩阵。

$$\text{而 } \delta_{ki} = \begin{cases} 1 & k=j \\ 0 & k \neq j \end{cases}$$

假设系统的初始状态 $X_0$ 也是高斯随机矢量,并且有:

$$E\{X_0\}=0$$

$$E\{X_0 X_0^*\}=p_0$$

并且还要假定 $X_0$ 和 $W_k$ 是相互独立的,即

$$E\{X_0 W_k^*\}=0 \quad k=0,1,2,\dots$$

这样就由(12—3—1)式确定了一个随机序列 $\{X_k\}$ ,  $k=1,2,\dots$ ; 为了进一步观察线性系统状态的演化,相继取 $k=j$ 和 $k=j+1$ ,得:

$$X_{i+1} = \Phi_{i+1,i} X_i + W_i$$

$$X_{i+2} = \Phi_{i+2,i+1} X_{i+1} + W_{i+1}$$

把第一式代到第二式中的 $X_{i+1}$ 的位置上得:

$$\begin{aligned} X_{i+2} &= \Phi_{i+2,i+1} (\Phi_{i+1,i} X_i + W_i) + W_{i+1} \\ &= \Phi_{i+2,i+1} \Phi_{i+1,i} X_i + \Phi_{i+2,i+1} W_i + W_{i+1} \quad (12-3-3) \end{aligned}$$

若令

$$\Phi_{i+p,i} = \Phi_{i+p,i+p-1} \Phi_{i+p-1,i+p-2} \cdots \Phi_{i+1,i} \quad (12-3-4)$$

则(12—3—4)式表现了系统从时刻 $j$ 到时刻 $j+p$ 的状态演化,那么可得出一般情况:

$$X_{i+p} = \Phi_{i+p,i} X_i + \sum_{l=i+1}^{i+p} \Phi_{i+p,l} W_{l-1} \quad (12-3-5)$$

令

$$\Phi_{kk} = I \quad k=1,2,\dots \quad (12-3-6)$$

当 $j=0$ 时,(12—3—5)式有:

$$X_p = \Phi_{p,0} X_0 + \sum_{i=1}^p \Phi_{p,i} W_{i-1} \quad (12-3-7)$$

(12—3—7)式表现了初态与随机干扰的传播情况。(12—3—5)式还表达了一个重要属性,即在给定 $X_1, X_2, \dots, X_i$ 之下,随机矢量 $X_p$ 的“条件分布”只依赖于 $X_i$ ,而不依赖于 $X_i$ 以前的诸状态 $X_{i-1}, \dots, X_1$ ,即以前的状态没有后效。这种无后效的属性叫做马尔可夫性。由(12—3—7)式看出,既然 $X_0$ 与 $W_{i-1}$  ( $i=1, \dots, p$ )都是高斯型的,而且高斯随机矢量的乘积与和仍都是高斯型的,所以 $X_p$ 也是高斯型的。 $\therefore$ 我们说, $\{X_p\}$ 是一个高斯-马尔可夫随机序列。

式(12—3—2)叫做量测方程,其中 $H_k$ 是量测矩阵,表达了从状态量 $X_k$ 到量测量 $Y_k$ 的转换矩阵,它的阶数是 $m \times n$ ,  $Y_k$ 是 $m$ 维量测矢量,  $V_k$ 是 $m$ 维的量测噪声。如前面所述,我们假设 $V_k$ 是高斯白噪声序列,其已知的均值和协方差矩阵为:

$$E\{V_k\}=0$$

$$E\{V_k V_k^*\}=R_k \delta_{ki}$$

这里 $R_k$ 是一个 $m \times m$ 维的正定矩阵,叫做量测噪声的协方差阵。又设 $V_k$ 和 $X_0$ 不相关的,即有

$$E\{X_0 V_k^*\}=0 \quad k=0,1,2,\dots$$

同时还假设  $W_k$  和  $V_k$  也是不相关的, 即

$$E\{W_k V_j^T\} = 0 \quad k, j = 0, 1, 2, 3, \dots$$

在以上所说明和假设下, 所谓滤波或估计就是指依据初值  $X_0$  与量测  $Y_1, Y_2, \dots, Y_k$ , 作出矢量  $X_k$  的按某种意义之下的“最好”的估计来。这是因为我们能量测的量只是系统的量测量  $Y_k$ , 而关心的却是系统的状态矢量  $X_k$ , 而且还要尽可能地排除随机干扰  $W_k$  和  $V_k$  的影响。

在开始时刻  $t_0$ , 对于  $X_0$  的估计取做  $E\{X_0\} = 0$ 。这个估计的误差协方差阵正是  $E\{X_0 X_0^T\} = P_0$ 。这里的初始估计的取法, 保证下面时刻的估计都是无偏的。怎样确定  $X_k$  的状态估计  $\hat{X}_k$  呢? 在这里我们以递推的形式表达出在每个时刻  $k$ , 在取得量测数据  $Y_1, \dots, Y_k$  的基础上作出状态  $X_k$  的估计  $\hat{X}_k$ 。所谓递推形式就是指确定  $\hat{X}_{k+1}$  的公式用前一时刻的估计  $\hat{X}_k$  与新量测值  $Y_{k+1}$  表达出来。既然  $\hat{X}_0$  已经确定, 那么当逐次令  $k=1, 2, \dots$  时就能用递推公式定出各估计值  $\hat{X}_1, \hat{X}_2, \dots$  来。

我们按下述做法求状态估计。假定根据量测数据  $Y_1, Y_2, \dots, Y_k$  已经确定出时刻  $k$  的状态  $X_k$  的估计  $\hat{X}_k$ 。在还没有取得新的量测数据  $Y_{k+1}$  之前, 要想估计下一时刻  $k+1$  的状态  $X_{k+1}$ , 我们只能从  $\hat{X}_k$  出发, 根据  $X_k$  的演化规律 (12-3-1) 式预报它。在 (12-3-1) 式中, 演化规律只是指  $X_{k+1} = \Phi_{k+1, k} X_k$ , 因为随机干扰  $W_k$  的值是事前不知道的。因此这种预报值只能取做:

$$\hat{X}'_{k+1} = \Phi_{k+1, k} \hat{X}_k \quad (12-3-8)$$

当取得了新的量测  $Y_{k+1}$  时, 利用这新的信息来检查预报值并根据检查结果修正它。检查的办法是根据量测方程 (12-3-2), 从对状态  $X_{k+1}$  的预报值  $\hat{X}'_{k+1}$  算出相应的量测预报值  $H_{k+1} \hat{X}'_{k+1}$ , 即有  $H_{k+1} \hat{X}'_{k+1} = H_{k+1} \Phi_{k+1, k} \hat{X}_k$  (因为这里并不知道量测噪声的值, 从而当作理想情况把它忽略)。然后把算出的量测值的预报值与真实的量测值  $y_{k+1}$  比较, 把比较的差值乘以适当的增益系数加在系统状态预报值  $\hat{X}'_{k+1}$  上修正它, 这就是在时刻  $k+1$  的估计值了。我们写成:

$$\hat{X}_{k+1} = \Phi_{k+1, k} \hat{X}_k + K_{k+1} (Y_{k+1} - H_{k+1} \Phi_{k+1, k} \hat{X}_k) \quad (12-3-9)$$

这里系数阵  $K_{k+1}$  是一个  $n \times m$  阶矩阵, 叫做增益矩阵。现在的问题是怎样确定这个增益矩阵。很自然的想法就是选取  $K_{k+1}$ , 使估计误差  $X_{k+1} - \hat{X}_{k+1}$  的协方差  $P_{k+1}$  的迹最小, 即:

$$\text{tr } P_{k+1} = E\{(X_{k+1} - \hat{X}_{k+1})(X_{k+1} - \hat{X}_{k+1})^T\} = \min$$

根据这个原则定出来的估计叫做极小方差递推滤波。由方程 (12-3-1) 和 (12-3-9), 再使用 (12-3-2) 式, 有:

$$X_{k+1} - \hat{X}_{k+1} = (I - K_{k+1} H_{k+1}) (\Phi_{k+1, k} (X_k - \hat{X}_k) + W_k) - K_{k+1} V_{k+1}$$

于是可由这个表达式算出:

$$\begin{aligned} P_{k+1} &= E\{(X_{k+1} - \hat{X}_{k+1})(X_{k+1} - \hat{X}_{k+1})^T\} \\ &= E\{[(I - K_{k+1} H_{k+1}) (\Phi_{k+1, k} (X_k - \hat{X}_k) + W_k) - K_{k+1} V_{k+1}] \cdot \\ &\quad \cdot [(I - K_{k+1} H_{k+1}) (\Phi_{k+1, k} (X_k - \hat{X}_k) + W_k) - K_{k+1} V_{k+1}]^T\} \end{aligned}$$

$$\begin{aligned}
&= (I - K_{k+1} H_{k+1}) \Phi_{k+1,k} E\{(X_k - \hat{X}_k)(X_k - \hat{X}_k)'\} \Phi_{k+1,k}' \\
&\cdot (I - K_{k+1} H_{k+1})' + (I - K_{k+1} H_{k+1}) E(W_k W_k') (I - K_{k+1} H_{k+1})' - \\
&- (I - K_{k+1} H_{k+1}) E\{(\Phi_{k+1,k} (X_k - \hat{X}_k) + W_k) V_{k+1}'\} K_{k+1}' - \\
&- K_{k+1} E\{V_{k+1} (\Phi_{k+1,k} (X_k - \hat{X}_k) + W_k)'\} (I - K_{k+1} H_{k+1})' + \\
&+ K_{k+1} E\{V_{k+1} V_{k+1}'\} K_{k+1}' \quad (12-3-10)
\end{aligned}$$

根据假设,  $W_k$  与  $V_{k+1}$  独立, 而  $X_k$  只依赖于  $X_0$  与  $W_0, W_1, \dots, W_{k-1}$ , 从而也和  $V_k$  独立. 又  $\hat{X}_k$  只依赖于  $\hat{X}_0, Y_1, \dots, Y_k$ , 从而只依赖于  $X_0, W_0, \dots, W_{k-1}, V_0, \dots, V_k$ , 所以也和  $V_{k+1}$  独立. 因此

$$E\{(\Phi_{k+1,k} (X_k - \hat{X}_k) + W_k) V_{k+1}'\} = 0$$

既然  $X_k - \hat{X}_k$  只依赖于  $X_0, W_0, \dots, W_{k-1}$ , 所以与  $W_k$  也独立, 所以

$$E\{(X_k - \hat{X}_k) W_k'\} = 0$$

于是 (12-3-10) 式能写成

$$\begin{aligned}
P_{k+1} &= (I - K_{k+1} H_{k+1}) (\Phi_{k+1,k} P_k \Phi_{k+1,k}' + Q_k) (I - K_{k+1} H_{k+1})' + \\
&+ K_{k+1} R_{k+1} K_{k+1}' \quad (12-3-11)
\end{aligned}$$

由于预报误差协方差阵  $P'_{k+1}$  等于

$$\begin{aligned}
&E\{(X_{k+1} - \hat{X}'_{k+1})(X_{k+1} - \hat{X}'_{k+1})'\} \\
&= E\{(\Phi_{k+1,k} (X_k - \hat{X}_k) + W_k)(\Phi_{k+1,k} (X_k - \hat{X}_k) + W_k)'\} \\
&= \Phi_{k+1,k} P_k \Phi_{k+1,k}' + Q_k
\end{aligned}$$

所以可以将 (12-3-11) 式写成:

$$\begin{aligned}
P_{k+1} &= (I - K_{k+1} H_{k+1}) P'_{k+1} (I - K_{k+1} H_{k+1})' + K_{k+1} R_{k+1} K_{k+1}' \\
&= P'_{k+1} - K_{k+1} H_{k+1} P'_{k+1} - P'_{k+1} H_{k+1}' K_{k+1}' + \\
&+ K_{k+1} (H_{k+1} P'_{k+1} H_{k+1}' + R_{k+1}) K_{k+1}' \quad (12-3-12)
\end{aligned}$$

不难导出,  $H_{k+1} P'_{k+1} H_{k+1}' + R_{k+1}$  是非负定阵, 因而从矩阵理论知道, 必存在可逆阵  $S$ , 使得

$$H_{k+1} P'_{k+1} H_{k+1}' + R_{k+1} = S S'$$

如用  $U$  表示  $P'_{k+1} H_{k+1}'$ , 那么 (12-3-12) 式可写成:

$$\begin{aligned}
P_{k+1} &= K_{k+1} S S' K_{k+1}' - K_{k+1} U - U K_{k+1}' + P'_{k+1} \\
&= (K_{k+1} S - U (S')^{-1}) (K_{k+1} S - U (S')^{-1})' + P'_{k+1} - U (S S')^{-1} U' \\
&= (K_{k+1} S - U (S')^{-1}) (K_{k+1} S - U (S')^{-1})' + P'_{k+1} - P'_{k+1} H_{k+1}' \\
&\cdot (H_{k+1} P'_{k+1} H_{k+1}' + R_{k+1})^{-1} H_{k+1} P'_{k+1} \quad (12-3-13)
\end{aligned}$$

既然 (12-3-13) 式右边最后两项与  $K_{k+1}$  无关, 而右边第一项是非负定的, 那么只要取  $K_{k+1}$  使这第一项为零, 就可使  $P_{k+1}$  的迹取极小值了. 为此只需取:  $K_{k+1} S - U (S')^{-1} = 0$

$$\text{即 } K_{k+1} = P'_{k+1} H_{k+1}' (H_{k+1} P'_{k+1} H_{k+1}' + R_{k+1})^{-1} \quad (12-3-14)$$

这时由 (12-3-13) 式得:

$$\begin{aligned}
P_{k+1} &= P'_{k+1} - P'_{k+1} H_{k+1}' (H_{k+1} P'_{k+1} H_{k+1}' + R_{k+1})^{-1} H_{k+1} P'_{k+1} \\
&= P'_{k+1} - K_{k+1} H_{k+1} P'_{k+1}
\end{aligned}$$

$$= (I - K_{k+1} H_{k+1}) P'_{k+1} \quad (12-3-15)$$

到此为止,就完成了极小方差递推滤波公式的全部推导介绍。现将结果摘要如下:

$$\hat{X}_{k+1} = \Phi_{k+1,k} \hat{X}_k + K_{k+1} (Y_{k+1} - H_{k+1} \Phi_{k+1,k} \hat{X}_k)$$

$$\text{且 } \hat{X}_0 = E\{X_0\} \quad (12-3-16)$$

$$K_{k+1} = P'_{k+1} H'_{k+1} (H_{k+1} P'_{k+1} H'_{k+1} + R_{k+1})^{-1} \quad (12-3-17)$$

$$P'_{k+1} = \Phi_{k+1,k} P_k \Phi'_{k+1,k} + Q_k \quad (12-3-18)$$

$P_0$  给定

$$P_{k+1} = (I - K_{k+1} H_{k+1}) P'_{k+1} \quad (12-3-19)$$

这样从  $\hat{X}_0$ ,  $P_0$  出发,利用已知的矩阵  $Q_k$ ,  $R_k$ ,  $H_k$ ,  $\Phi_{k+1,k}$ , 以及  $k$  时刻的量测  $Y_k$ , 就能递推地算出每个时刻的状态估计。

如果线性系统 (12-3-1) 和 (12-3-2) 是定常的, 即矩阵  $\Phi_{k+1,k} = \Phi$ ,  $H_{k+1} = H$ , 都是常阵, 模型噪声  $W_k$  和量测噪声  $V_k$  都是平稳随机序列, 即  $Q_k$  和  $R_k$  都是常阵, 那么对任意  $P_0 \geq 0$ , 都有:

$$\begin{aligned} \lim_{k \rightarrow \infty} P(t_k, t_0, P_0) &= P > 0 \\ \lim_{k \rightarrow \infty} K_k &= K \\ \lim_{k \rightarrow \infty} P'_k &= P' > 0 \end{aligned}$$

$\therefore$  常增益的离散 Kalman 滤波器是渐近稳定的。

## 二、程序说明

本程序完全是按上述算法编制的, 使用了 A、C、W、V、P、U、F、Q、R、Y、Z、Q 等数组, 并使用 N、M、N8、N9、S8、S9、S、K<sub>1</sub>、K<sub>2</sub>、T 等简单变量。

整个程序大致分为三个部分。第一部分为输入已知数据部分, 由 8000—8116 句组成; 第二部分为迭代滤波过程, 由 8118—8370 句组成; 第三部分印出增益矩阵 K 和协方差阵, 由 8372—8406 语句组成。

在迭代滤波过程中, 每次迭代之前先由 (12-3-16) 算出本次的  $X(I)$  各状态值并印出, 这时使用的是上次迭代计算中求出的 K 阵之值。为了观察滤波的过度过程, 并把本程序所带的例题中系统应该得到的状态分量真值印出, 以便与滤波后得到的状态分量进行比较。同时还将每次得到的新息与旧息的差值印出来。

为验证滤波对噪声抑制的能力, 在每次采样新息时都加入了一个标准正态分布的伪随机数, 即  $N(0, 0.25)$ 。有关伪随机码产生技术请参看本书第六章的内容。这部分程序就是 8434—8512 语句。

## 三、程序使用

若使用本程序所带例题, 则无须改动。若使用者计算另外的题目, 要改 8450 和 8482 两句, 前一句为量测新值, 后一句为系统状态变量真值印出语句。

采样间隔定为  $T = 0.04$  秒, 若要改动请改 8124 语句; 正态分布的伪随机数取  $\mu = 0, \sigma^2 = 0.25$  若改成其他值, 请改 8510 句。

具体上机操作如下:

1. 用BASIC将本程序引入机内, 然后按上述将要改动的语句改好, 用RUN/键盘命令启动本程序, 机器将自动印出: "SYSTEM DIMENSION N?" 询问系统的阶数。

2. 用键盘回答系统阶数后, 机器又自动印出: "MEASURE DIMENSION?" 询问量测维数。

3. 用键盘回答量测维数后, 机器又自动印出: "INPUT INITIAL STATE X(1)", 要求输入系统的初始状态值。

4. 用键盘回答系统的初始状态值后, 机器又自动印出: "INPUT MATRIX F(I,J)", 要求输入转移矩阵各元素之值。

5. 用键盘按行回答转移矩阵各元素之值后, 机器又自动印出: "MEASUREMENT MATRIX C", 要求输入系统的量测矩阵。

6. 用键盘按行回答量测矩阵各元素之值后, 机器又自动印出: "DISTURBANCE MATRIX Q", 要求输入模型噪声协方差阵Q的各元素之值。

7. 用键盘按行回答模型噪声阵Q后, 机器又自动印出: "MEASUREMENT ERROR MATRIX R" 要求输入量测误差的协方差阵R的各元素之值。

8. 用键盘回答量测噪声阵R后, 机器又自动印出: "INITIAL COVARIANCE MATRIX P<sub>0</sub>", 要求输入初始的估计误差协方差阵P<sub>0</sub>的各元素之值。

9. 用键盘回答P<sub>0</sub>阵之值后, 机器又自动印出: "ITERATIONAL NUMBER N<sub>0</sub>", 要求输入迭代滤波计算的迭代次数。

10. 用键盘回答N<sub>0</sub>之值后, 机器开始进行迭代运算, 每增加一个步距, 印出时间 T 之值和系统状态的估计值和真值, 然后印出时间 T 的量测与估计值之差值, 直到要求的迭代次数全部作完, 再印出稳定增益阵K之值和当时的估计误差协方差阵 P 的值。最后印出: "C=1 CONTINUE C=0 STOP", 询问是否还要继续进行迭代运算。若要继续迭代运算, 则用键盘回答 1; 否则回答 0。

11. 当回答 0 时, 机器印出总的迭代次数之后, 停止运行。当回答 1 时, 机器又印出: "ADDITIONAL ITERATIONAL NUMBER", 询问迭代次数再增加多少。待使用者回答后又按(10)那样运行, 边印出各种信息。

#### 四、例题

已知 信号源运动方程为:

$$S(t) = 5 - 2t + 3t^2 + W(t)$$

$$V(t) \text{ 为一个正态白噪声 } \mathcal{N}(0, 0.25)$$

若用差分方程来描述这个系统, 则:

$$X_k = \begin{pmatrix} S \\ \dot{S} \\ \ddot{S} \\ S \end{pmatrix} \quad H_k = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Phi_{k,k-1} = \begin{pmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \quad T \text{ 为采样间隔}$$

$$\text{则有 } \begin{cases} X_k = \Phi_{k,k-1} X_{k-1} + W_{k-1} \\ Y_k = H_k X_k + V_k \end{cases}$$

$$k = 1, 2, \dots$$

对于这样系统取系统阶数 $N=3$ , 量测维数 $M=1$

## 五、程序清单和运行结果

(见后附)

## 六、就这个例题我们做过以下试验

$\because V(t)$ 为 $\mathcal{N}(0, 0.25)$

$\therefore$ 取 $R=0.25$

而 $Q$ 阵取了6组, 现将6组计算的数据摘录几点抄在下面。迭代次数 $N_0=50$ ,  $T=0.04$

第一组, 取

$$Q = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.001 \end{pmatrix} \quad P_0 = 0$$

$$\text{则得出 } K = \begin{pmatrix} 0.064 \\ 0.0084 \\ 0.00607 \end{pmatrix} \quad P = \begin{pmatrix} 0.0016 & 0.0021 & 0.0015 \\ 0.0021 & 0.0032 & 0.00278 \\ 0.0015 & 0.00278 & 0.0036 \end{pmatrix}$$

第二组, 取

$$Q = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \quad P_0 = 0$$

$$\text{则得出 } K = \begin{pmatrix} 0.228 \\ 0.7373 \\ 0.5148 \end{pmatrix} \quad P = \begin{pmatrix} 0.0057 & 0.184 & 0.129 \\ 0.184 & 1.184 & 0.911 \\ 0.129 & 0.911 & 3.205 \end{pmatrix}$$

第三组, 取

$$Q = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}$$

$$\text{则得出 } K = \begin{pmatrix} 0.24 \\ 0.849 \\ 0.859 \end{pmatrix} \quad P = \begin{pmatrix} 0.60 & 0.21 & 0.215 \\ 0.21 & 1.398 & 1.62 \\ 0.215 & 1.62 & 6.08 \end{pmatrix}$$

第四组, 取

$$Q = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}$$

$$\text{则得出 } K = \begin{pmatrix} 0.49 \\ 0.70 \\ 0.36 \end{pmatrix} \quad P = \begin{pmatrix} 0.1235 & 0.1755 & 0.091 \\ 0.1755 & 4.1410 & 2.21 \\ 0.091 & 2.21 & 3.95 \end{pmatrix}$$

第五组, 取

$$Q = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}$$



$$\text{则得出 } K = \begin{pmatrix} 0.5 \\ 1.032 \\ 0.61 \end{pmatrix} \quad P = \begin{pmatrix} 0.1269 & 0.258 & 0.15 \\ 0.258 & 6.33 & 3.845 \\ 0.15 & 3.845 & 9.04 \end{pmatrix}$$

第六组, 取

$$Q = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{则得出 } K = \begin{pmatrix} 0.2757 \\ 1.108 \\ 1.70 \end{pmatrix} \quad P = \begin{pmatrix} 0.068 & 0.277 & 0.425 \\ 0.277 & 1.975 & 3.72 \\ 0.425 & 3.72 & 16.278 \end{pmatrix}$$

其他中间信息摘录如下:

状态 组号	时 真 值 间 估 值	1.04	1.36	1.68	1.96
		6.16, 4.24, 6	7.8, 6.16, 6	10.10, 8.08, 6	12.6, 9.75, 6
1		5.00, 0.0007, 0.0009 0.55	5.05, 0.11, 0.11 3.04	5.2, 0.37, 0.30 5.65	5.5, 0.68, 0.48 8.667
2		5.04, 0.13, 0.006 1.599	5.46, 1.46, 0.87 2.066	6.03, 3.3, 2.2 4.545	6.89, 6.10, 4.2 8.29
3		5.38, 1.3, 1.15 1.68	5.64, 2.22, 2.15 2.67	6.04, 3.6, 3.6 4.28	7.06, 7.17, 7.2 8.45
4		5.4, 0.398, 0.15 0.921	6.30, 1.46, 0.66 2.68	7.46, 3.2, 1.58 5.0	8.71, 5.24, 2.7 7.53
5		5.44, 0.70, 0.29 0.88	6.0, 1.8, 0.91 1.996	7.39, 4.66, 2.6 4.7	8.84, 7.78, 4.58 7.57
6		5.62, 2.5, 3.85 2.3	5.8, 3.3, 5.07 2.989	6.3, 5.29, 8.11 4.78	7.05, 8.23, 12.6 7.43

#### 程序清单

```

8000 REM          KAEMAN FIETER
8002 REM          PARAMETERS
8004 PRINT "SYSTEM DIMENSION N";
8006 INPUT N
8008 PRINT
8010 PRINT "MEASURE DIMENSION";
8012 INPUT M
8014 PRINT
8016 DIM A(N,N),C(M,N),W(N,N),V(M,M),P(N,N)U(N,N),F(N,N)
8018 PRINT "INPUT INITIAL STATE X(1)"
8020 FOR I=1 TO N
8022   INPUT X(I)
8024 NEXT I

```

```

8026 PRINT
8028 PRINT "INPUT MATRIX F(I,J)"
8030 FOR I=1 TO N
8032   FOR J=1 TO N
8034     INPUT F(I,J)
8036   NEXT J
8038 PRINT
8040 NEXT I
8042 REM           MATRIX C
8044 PRINT
8046 PRINT "MEASUREMEN MATRIX C"
8048 FOR I=1 TO M
8050   FOR J=1 TO N
8052     INPUT C(I,J)
8053   NEXT J
8054 PRINT
8055 NEXT I
8056 REM           MATRIX Q
8058 PRINT
8060 PRINT "DISTURBANCE MATRIX Q"
8062 FOR I=1 TO N
8064   FOR J=1 TO N
8066     INPUT Q(I,J)
8068   NEXT J
8070 PRINT
8072 NEXT I
8074 PRINT
8076 PRINT "MEASUREMENT ERROR MATRIX R"
8078 FOR I=1 TO M
8080   FOR J=1 TO M
8082     INPUT R(I,J)
8084   NEXT J
8086 PRINT
8088 NEXT I
8090 REM   MATRIX P
8092 PRINT
8094 PRINT "INITIAL COVARANCE MATRIX P0"
8096 FOR I=1 TO N
8098   FOR J=1 TO N
8100     INPUT P(I,J)

```

```

8102     NEXT J
8104     PRINT
8106     NEXT I
8110     PRINT "ITERATIONAL NUMBER N9"
8112     INPUT N9
8114     PRINT
8116     LET N8=1
8118     REM    LOOP CALCVLATION
8120     FOR I9=N8 TO N9
8122         GOTO 8434
8124         LET S9=S9+.04
8126     REM          U=F * P * TR(F)
8128         FOR I=1 TO N
8130             FOR J=1 TO N
8132                 LET U(I,J)=0
8134                 FOR K2=1 TO N
8136                     LET S=0
8138                     FOR K1=1 TO N
8140                         LET S=S+F(I,K1) * P(K1,K2)
8142                     NEXT K1
8144                         LET U(I,J)=U(I,J)+S * F(J,K2)
8146                     NEXT K2
8148                 NEXT J
8150             NEXT I
8152     REM          U=U+Q
8154         FOR I=1 TO N
8156             FOR J=1 TO N
8158                 LET U(I,J)=U(I,J)+Q(I,J)
8160             NEXT J
8162         NEXT I
8164     REM    Y=C * U * TR(C)
8166         FOR I=1 TO M
8168             FOR J=1 TO M
8170                 LET Y(I,J)=0
8172                 FOR K2=1 TO N
8174                     LET S=0
8176                     FOR K1=1 TO N
8178                         LET S=S+C(I,K1) * U(K1,K2)
8180                     NEXT K1
8182                         LET Y(I,J)=Y(I,J)+S * C(J,K2)

```

```

8184             NEXT K2
8186             NEXT J
8188             NEXT I
8190 REM       Y=Y+R
8192             FOR I=1 TO M
8194                 FOR J=1 TO M
8196                     LET Y(I,J)=Y(I,J)+R(I,J)
8198                 NEXT J
8200             NEXT I
8202 REM       Z=INV(Y)
8204             FOR I=1 TO M
8206                 FOR J=1 TO M
8208                     LET Z(I,J)=0
8210                 NEXT J
8212             LET Z(I,I)=1
8214             NEXT I
8216             FOR J=1 TO M
8218                 FOR I=1 TO M
8220                     IF Y(I,J)><0 THEN GOTO 8228
8222                 NEXT I
8224                 PRINT "SINGULAR"
8226                 STOP
8228                 FOR K=1 TO M
8230                     LET S=Y(J,K)
8232                     LET Y(J,K)=Y(I,K)
8234                     LET Y(I,K)=S
8236                     LET S=Z(J,K)
8238                     LET Z(J,K)=Z(I,K)
8240                     LET Z(I,K)=S
8242                 NEXT K
8244                 LET T=1/Y(J,J)
8246                 FOR K=1 TO M
8248                     LET Y(J,K)=T * Y(J,K)
8250                     LET Z(J,K)=T * Z(J,K)
8252                 NEXT K
8254                 FOR L=1 TO M
8256                     IF L=J THEN GOTO 8268
8258                     LET T=-Y(L,J)
8260                     FOR K=1 TO M
8262                         LET Y(L,K)=Y(L,K)+T * Y(J,K)

```

```

8264             LET Z(L,K)=Z(L,K)+T * Z(J,K)
8266             NEXT K
8268         NEXT L
8270     NEXT J
8272 REM     Y=U * TR(C)
8274     FOR I=1 TO N
8276         FOR J=1 TO M
8278             LET Y(I,J)=0
8280             FOR K=1 TO N
8282                 LET Y(I,J)=Y(I,J)+U(I,K) * C(J,K)
8284             NEXT K
8286         NEXT J
8288     NEXT I
8290 REM     O=Y * Z(O=KALMANGAIN)
8292     FOR I=1 TO N
8294         FOR J=1 TO M
8296             LET O(I,J)=0
8298             FOR K=1 TO M
8300                 LET O(I,J)=O(I,J)+Y(I,K) * Z(K,J)
8302             NEXT K
8304         NEXT J
8306     NEXT I
8308 REM     Y=O * C
8310     FOR I=1 TO N
8312         FOR J=1 TO N
8314             LET Y(I,J)=0
8316             FOR K=1 TO M
8318                 LET Y(I,J)=Y(I,J)+O(I,K) * C(K,J)
8320             NEXT K
8322         NEXT J
8324     NEXT I
8326 REM     Z=I
8328     FOR I=1 TO N
8330         FOR J=1 TO N
8332             LET Z(I,J)=0
8334         NEXT J
8336         LET Z(I,I)=1
8338     NEXT I
8340 REM     Y=Z-Y
8342     FOR I=1 TO N

```

```

8344             FOR J=1 TO N
8346             LET Y(I,J)=Z(I,J)-Y(I,J)
8348             NEXT J
8350         NEXT I
8352     REM     P=Y*U
8354     FOR I=1 TO N
8356         FOR J=1 TO N
8358             LET P(I,J)=0
8360             FOR K=1 TO N
8362                 LET P(I,J)=P(I,J)+Y(I,K)*U(K,J)
8364             NEXT K
8366         NEXT J
8368     NEXT I
8370 NEXT I9
8372 REM     OUTPUT MATRIX K
8374 PRINT
8376 PRINT "GAIN MATRIX K"
8378 FOR I=1 TO N
8380     FOR J=1 TO M
8382         PRINT TAB(10*(J-1)); O(I,J);
8384     NEXT J
8386     PRINT
8388 NEXT I
8390 REM     OUTPUT MATRIX P
8392 PRINT
8394 PRINT "COVARIANCE MATRIX P"
8396 FOR I=1 TO N
8398     FOR J=1 TO N
8400         PRINT TAB(10*(J-1))P(I,J);
8402     NEXT J
8404     PRINT
8406 NEXT I
8408 PRINT "C=1 CONTINUE C=0 STOP"
8410 INPUT C
8412 PRINT
8414 IF C=0 THEN GOTO 8428
8416 PRINT "ADDITIONAL ITERATIONAL NUMBER"
8418 INPUT N7
8420 PRINT
8422 LET N8=N9+1

```

```

8424 LET N9=N9+N7
8426 GOTO 8118
8428 PRINT "TOTAL ITERATIVE NUMBER=",N9
8430 STOP
8432 END
8434 FOR I=1 TO N
8436   LET U(I)=0
8438   FOR J=1 TO N
8440     LET U(I)=U(I)+F(I,J)*X(J)
8442   NEXT J
8444 NEXT I
8446 FOR I=1 TO M
8448   GOSUB 8502
8450   LET H(I)=5-2*S9+3*S9*S9+S8
8452 NEXT I
8454 FOR I=1 TO M
8456   LET G(I)=0
8458   FOR J=1 TO N
8460     LET G(I)=G(I)+C(I,J)*U(J)
8462   NEXT J
8464   LET H(I)=H(I)-G(I)
8466 NEXT I
8468 PRINT "T=",S9,"X(I):"
8470 FOR I=1 TO N
8472   LET G(I)=0
8474   FOR J=1 TO N
8476     LET G(I)=G(I)+O(I,J)*H(J)
8478   NEXT J
8480   LET G(I)=U(I)+G(I)
8482   PRINT G(I),
8484 NEXT I
8486 PRINT
8488 PRINT TAB(5);5-2*S9+3*S9*S9,-2+6*S9,6
8490 PRINT TAB(5);"Y(I):",
8492 FOR I=1 TO M
8494   PRINT H(I),
8496 NEXT I
8498 PRINT
8500 GOTO 8124
8502 LET S8=0

```

```

8504 FOR J=1 TO 12
8506 LET S8=S8+RND(0)
8508 NEXT J
8510 LET S8=0.5*(S8-6)
8512 RETURN

```

运行结果

```

SYSTEM DIMENSION N? 3
MEASURE DIMENSION? 1
INPUT INITIAL STATE X(I)
? 5? 0? 0
INPUT MATRIX F(I,J)
? 1 ? 0.04 ? 0.0008
? 0 ? 1 ? 0.04
? 0 ? 0 ? 1
MEASUREMEN MATR IX C
? 1 ? 0 ? 0
DISTURBANCE MATR IX Q
? 0 ? 0 ?
? 0 ? 0.1 ? 0
? 0 ? 0 ? 0.1
MEASUREMENT ERROR MATRIX R
? 0.25
INITIAL COVARIANCE MATRIX P0
? 0 ? 0 ? 0
? 0 ? 0 ? 0
? 0 ? 0 ? 0
ITERATIONAL NUMBER N9
? 50
T= 0 X(I);
5 0 0.....滤波估计出来的状态各值
5 -2 6.....计算出来的状态各真值
Y(I);-.133697.....量测新值与估计值之差值
T= .04 X(I);
5 0 0
4.9248 -1.76 6
Y(I); 3.10955E-2
T= .08 X(I);
5.00019 4.66366E-3 9.31987E-5
4.8592 -1.52 6
Y(I); .291432

```



T= .12 X(I);  
 4 .99967 -4.92416E-3 -1.63774E-4  
 4.8032 -1.28 6

Y(I);-.102713

T= .16 X(I);  
 5 .00531 5.70751E-2 2.6532E-3  
 4.7568 -1.04 6

Y(I); .59871

T= .2 X(I);  
 5 .01208 .101163 6.03848E-3  
 4.72 -.8 6

Y(I); .645634

T= .24 X(I);  
 4 .99294 -4.85017E-2 -3.53778E-3  
 4.6928 -.56 6

Y(I);-.211876

:

T= 1.76 X(I);  
 6 .35924 4.37475 2.94801  
 10.7728 8.55999 6

Y(I); 5.99553

T= 1.8 X(I);  
 6 .33094 4.28754 2.91005  
 11.12 8.79999 6

Y(I);5.86398

T= 1.84 X(I);  
 6 .55138 5.00192 3.41747  
 11.4768 9.03999 6

Y(I); 6.82796

T= 1.88 X(I);  
 6 .6233 5.23799 3.60067  
 11.8432 9.27999 6

Y(I); 7.13747

T= 1.92 X(I);  
 6 .60317 5.17693 3.57879  
 12.2192 9.51999 6

Y(I); 7.04254

T= 1.96 X(I);  
 6 .88946 6.10569 4.24281  
 12.6048 9.75999 6

Y(I), 8.29311  
 GAIN MATRIX K  
   .228016  
   .7373  
   .514804  
 COVARIANCE MATRIX P  
   5.70039E-2      .184325    .128701  
   .184325          1.18446    .911228  
   .128701          .911228    3.20494  
 C=1 CONTINUE C=0 STOP  
   ? 1  
 ADDITIONAL ITERATIONAL NUMBER  
   ? 15  
 T= 2 X(I),  
   6 .96988    6.36971    4.44752  
      13    9.99999      6  
      Y(I), 8.63924  
 T= 2.04 X(I),  
   6 .90098    6.15064    4.31357  
      13.4048    10.24    6  
      Y(I), 8.33092  
 T= 2.08 X(I),  
   7 .08203    6.74023    4.74634  
      13.8192    10.48      6  
      Y(I), 9.11811  
      :  
 T= 2.36 X(I),  
   7.81953    9.15442    6.5825  
      16.9888    12.16    6  
      Y(I), 12.3036  
 T=2.4 X(I),  
   7.8324    9.19894    6.62863  
      17.48    12.4      6  
      Y(I), 12.3551  
 T=2.44 X(I),  
   8.15375    10.2454    7.39727  
      17.9808    12.64    6  
      Y(I), 13.7521  
 T=2.48 X(I),  
   8.15306    10.2458    7.41097

```

      18.4912 12.88    6
      Y(I), 13.7447
T= 2.52 X(I),
      8.11684 10.1305 7.33985
      19.0112 13.12    6
      Y(I), 13.5828
T= 2.56 X(I),
      8.332 10.8321 7.86036
      19.5408 13.36    6
      Y(I), 14.5165
GAIN MATRIX K
      .22959
      .746535
      .542497
COVARIANCE MATRIX P
      5.73975E-2 .186634 .135624
      .186634 1.19803 .951833
      .135624 .951832 3.32726
C=1 CONTINUE C=0 STOP
? 0
TOTAL ITERATIVE NUMBER= 65
STOP AT 8430

```

#### 参考文献:

- (1)中国科学院数学研究所 王恩平  
《卡尔曼滤波器初等介绍》
- (2)中国科学院数学研究所  
《离散时间系统滤波的数学方法》

1977.7北航学习班

国防工业出版社 1975.9

## 第十三章 信号分析

### §1 直接法进行富里叶分解

#### 一、计算方法概要

连续函数  $x(t)$  的富里叶级数为:

$$x(t) = a_0 + 2 \sum_{K=1}^{\infty} \left( a_K \cos \frac{2\pi Kt}{T} + b_K \sin \frac{2\pi Kt}{T} \right)$$

$$\text{其中 } a_0 = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) dt$$

$$a_K = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) \cos \frac{2\pi Kt}{T} dt \quad (13-1-1)$$

$$b_K = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) \sin \frac{2\pi Kt}{T} dt \quad K=1, 2, \dots, M$$

所谓直接方法进行富里叶分解, 就是在  $x(t)$  为已知时, 直接使用辛普生数值积分公式求 (13-1-1) 式中各项积分, 求出相应的  $a_K, b_K$  之值。辛普生积分公式为:

$$\int_A^B f(x) dx = \frac{h}{3} (y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \dots + 2y_{N-2} + 4y_{N-1} + y_N) \quad (13-1-2)$$

有关辛普生积分详细内容请参看本书第九章。

#### 二、程序说明

本程序只在  $0-2\pi$  区间对函数进行富里叶分解的。在  $0-2\pi$  之间要分多少间隔, 由使用者决定; 另外, 富里叶分解出的谐波数也是由使用者给出的。

本程序使用数组  $A(I)$ ,  $B(I)$ , 和  $Y(I)$ 。使用的简单变量有  $N, M, D_1 \sim D_4, S_1 \sim S_4$  等。

程序结束原始信息没有破坏, 分解出的各谐波的实部在  $A(K)$  的组中, 虚部在  $B(K)$  数组中。

#### 三、程序使用

1. 由BASIC引入本程序, 并在语句标号为8000句中, 用DEF定义将要进行富里叶分解的时间函数。

例如:  $x(t) = \sin(2t) - (\cos(3t))^2$  如下定义:

8000 DEF FNC(X) = SIN(2 \* X) + COS(3 \* X) \* COS(3 \* X)

2. 用键盘命令 RUN 启动本程序, 机器自动印出: "INPUT NO. OF INTERVAL N", 询问在  $0-2\pi$  之间分多少间隔。
3. 用键盘回答N值之后, 机器又自动印出: "INPUT MAXIMUM HARMONIC

TO BE CALCULATED M ( $<N$ )"询问要求分解得到的最大谐波次数M。

4. 用键盘回答M之值后, 机器立即印出给定时间函数在间隔点的函数值及曲线。接着印出分解好的各个谐波的系数 $a_k$ ,  $b_k$ 之值, 印出格式为:

HARMONTIC            COSINE COEFF.            SINE COEFF.

\* \* \*  
:

\* \* \*  
:

\* \* \*  
:

5. 若 $x(t)$ 为确定函数如上操作; 若 $x(t)$ 为列表函数则将8020句改为INPUT Y(I), 即表列函数值由键盘输入到数组Y中。

#### 四. 程序清单及运行结果

##### 程序清单

```

8000 DEF FNC(X)=SIN(2 * X)+COS(3 * X) * COS(3 * X)
8002 PRINT "INPUT NO. OF INTERVAL N"
8004 INPUT N
8006 PRINT
8008 PRINT "INPUT MAXIMUM HARMONIC TO BE CALCULATED M(<N)"
8010 INPUT M
8012 PRINT
8014 DIM A(M),B(M),Y(N)
8016 LET D1=2 * 3.14159/N
8018 FOR I=0 TO N-1
8020     LET Y(I)=FNC(I * D1)
8021     PRINT Y(I); TAB (25+10 * Y(I)+.5); "."
8022 NEXT I
8023 PRINT
8024 PRINT "HARMONIC","COSINE COEFF.", "SINE COEFF."
8026 FOR K=0 TO M
8028     LET D2=D1 * K
8030     LET D3=0
8032     LET D4=0
8034     LET D5=0
8036     LET D6=0
8038     LET S1=Y(0)
8040     LET S2=Y(N-1) * COS((N-1) * D2)
8042     LET S3=Y(N-1) * SIN((N-1) * D2)
8044     FOR I=1 TO N-2
8046         IF I/2=INT(I/2) GOTO 8054
8048         LET D3=D3+Y(I) * COS(I * D2)
8050         LET D5=D5+Y(I) * SIN(I * D2)
8052         GOTO 8058
8054         LET D4=D4+Y(I) * COS(I * D2)

```

```

8056      LET D6=D6+Y(I)*SIN(I*D2)
8058      NEXT I
8060      LET A(K)=(D1/3*(S1+S2+4*D3+2*D4))/(2*3.14159)
8062      LET B(K)=(D1/3*(S3+4*D5+2*D6))/(2*3.14159)
8066      PRINT K,A(K),B(K)
8068      NEXT K
8070      END

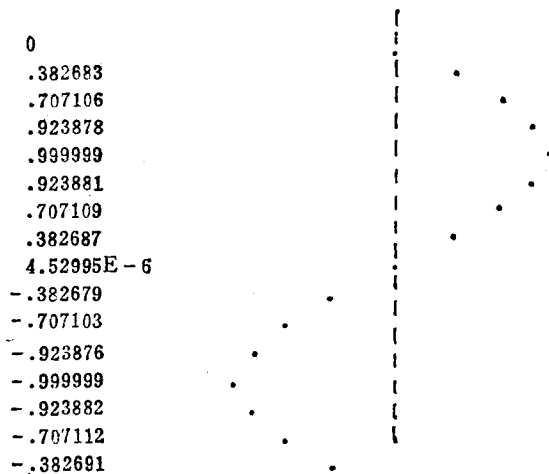
```

第一次运行结果

```

8000 DEF FNC(X)=SIN(X)
RUN
INPUT NO. OF INTERVAL N
? 16
INPUT MAXIMUM HARMONIC TO BE CALCULATED M(<N)
? 8

```



HARMONIC	COSINE COEFF.	SINE COEFF.
0	2.39183E-2	0
1	2.20983E-2	.490847
2	1.69125E-2	-.016913
3	9.1527E-3	-2.20973E-2
4	-5.05836E-7	-.023918
5	-9.15395E-3	-2.20969E-2
6	-1.69135E-2	-.016911
7	-2.20921E-2	.157514
8	-2.39179E-2	-2.67172E-6

END AT 8070

第二次运行结果

8000 DEF FNC(X)=COS(X)

RUN

INPUT NO. OF INTERVAL N

? 16

INPUT MAXIMUM HARMONIC TO BE CALCULATED M(<N)

? 8

.999999  
 .92388  
 .707108  
 .382686  
 3.21865E-6  
 -.38268  
 -.707104  
 -.923877  
 -.999999  
 -.923882  
 -.707112  
 -.382691  
 -8.22544E-6  
 .382675  
 .707101  
 .923875

HARMONIC	COSINE COEFF.	SINE COEFF.
0	-7.85769E-2	0
1	.425819	2.20983E-2
2	-6.16638E-2	4.08308E-2
3	-4.29306E-2	5.33477E-2
4	-2.08328E-2	5.77427E-2
5	1.26419E-3	5.33468E-2
6	1.99957E-2	4.08292E-2
7	-.134152	2.21012E-2
8	3.69094E-2	-2.12813E-6

END AT 8070

第三次运行结果

改: 8000 DEF FNC(X)=SIN(2 \* X)+COS(3 \* X) \* COS(3 \* X)

RUN

INPUT NO. OF INTERVAL N

? 16

INPUT MAXIMUM HARMONIC TO BE CALCULATED M(<N)

? 8

```

.999998
.853555
1.5
1.56067
4.53002E-6
.146442
-.499987
-.560673
.99999
.853558
1.49998
1.56068
1.28751E-5
.146437
-.499975
-.56069

```

HARMONIC	COSINE COEFF.	SINE COEFF.
0	.514208	0
1	1.15403E-2	-1.34103E-2
2	-7.93876E-2	.475224
3	-7.42322E-3	-3.23753E-2
4	-2.08349E-2	-3.50423E-2
5	-.034245	-3.23731E-2
6	.204392	.14189
7	-5.32099E-2	-1.34102E-2
8	-.22254	7.08028E-6

END AT 8070

## §2 直接法求相关函数

### 一、计算方法概要

相关函数分为自相关和互相关两种。自相关就是在一个数据序列  $\{x_k\}$ ,  $k=1, 2, \dots, n$  上进行相关运算所得到的新的数据序列  $R_x(r)$ ,  $r=0, 1, \dots, n-1$ ; 而互相关就是在两个数据序列  $\{x_k\}, \{y_k\}$ ,  $k=1, 2, \dots, n$  上进行相关运算所得到的新的数据序列  $R_{xy}(r)$ ,  $r=0, 1, \dots, n-1$ 。

相关函数是由以下两式确定的:

自相关函数为:

$$R_x(r) = \frac{1}{N-r} \sum_{i=0}^{N-r} x(i)x(i+r) \quad (13-2-1)$$

互相关函数为:

$$R_{xy}(r) = \frac{1}{N-r} \sum_{i=0}^{N-r} x(i)y(i+r) \quad (13-2-2)$$

相关函数在许多工程领域中都有广泛应用, 比如在机械系统的振动分析, 通讯、雷达定



向 控制系统辨识以及数据分析检索等等数据信号处理中都广泛应用。

计算相关函数 有两种方法：即直接法和间接法。计算相关函数的间接方法就是用快速富里叶方法（见本章第三节内容）。而计算相关函数的直接方法就是使用定义式（13—2—1）和（13—2—2）式。

## 二、程序说明

本程序既可以求自相关，也可以求互相关。在求互相关函数时，原始数据序列分别存放在A(I)和B(I)两数组中，求出的相关序列放在C(I)数组之中。若求自相关时，原始序列同时放在两个数组A(I)和B(I)之中，求出的有相关序列放在C(I)数组之中。

本程序中除使用以上数组之外，还使用N、R、K等简单变量。计算结束，A(I)和B(I)中的原始数据保持不变。在我们后面所附的程序，若将8016句去掉，还可以印出原始数据序列及曲线。

## 三、程序使用

1. 本程序中所计算的是求[0, 1]之间的均匀分布的随机数RND(0)的自相关。使用者若要求另外的数据序列的相关函数，需要改动8014句和8028句。这两个语句是分别给数组A(I)和B(I)输入原始数据的。根据使用者的需要，可以改为另外的赋值语句、键盘输入语句或读数据区语句皆可。
2. 由BASIC解释程序引入本程序，按（1）说明改好源程序，然后用键盘命令RUN启动本程序，则机器自动印出：“INPUT N”，询问原始数据的最大个数。
3. 由键盘回答数据个数后，若要求打印原始数据序列去掉8016句，则机器首先印出原始数据序列，然后进行相关运算，并印出相关数据序列。

四、试题：求[0, 1]之间的均匀分布的随机变量X的自相关。

$$\because \text{密度函数 } f(x) = \begin{cases} 1 & \text{当 } 0 \leq x \leq 1 \\ 0 & \text{其它} \end{cases}$$

$$\text{则有: } E(x) = \int_{-\infty}^{+\infty} xf(x)dx = \int_0^1 xdx = \frac{1}{2}\mu$$

$$D(x) = \int_0^1 (x - \frac{1}{2})^2 dx = \frac{1}{12}\sigma^2$$

$$\text{而 } \Psi^2(x) = \int_0^1 x^2 dx = \mu^2 + \sigma^2 = R_x(0) = 1/3$$

## 五、程序清单与运行结果

### 程序清单

```
8000 PRINT "INPUT N"
8002 INPUT N
8004 PRINT
8006 LET R=N-1
8008 DIM A(2*N),B(2*N),C(2*N)
8010 PRINT "A(I): "
8012 FOR I=1 TO N
8014 LET A(I)=RND(0)
8016 GOTO 8020
```

```

8018 PRINT I-1; TAB(5); A(I); TAB(40+10 * A(I)); "+"
8020 NEXT I
8022 PRINT
8024 PRINT "B(I); "
8026 FOR I=1 TO N
8028 LET B(I)=A(I)
8030 NEXT I
8032 PRINT "C(K); "
8034 FOR K=0 TO R
8036 FOR I=1 TO N-K
8038 LET C(K)=C(K)+A(I) * B(I+K)
8040 NEXT I
8042 LET C(K)=C(K)/(N-K)
8044 IF K<10 THEN GOTO 8050
8046 IF K>N-10 THEN GOTO 8050
8048 GOTO 8052
8050 PRINT K; TAB(6); C(K); TAB(40+10 * C(K)); "."
8052 NEXT K
8054 END

```

运行结果

RUN

INPUT N

? 1500

A(I);

B(I);

C(K);

0	.338261	.
1	.257007	.
2	.253541	.
3	.256687	.
4	.253065	.
5	.249879	.
6	.258883	.
7	.25538	.
8	.257572	.
9	.255991	.
10	.254631	.
20	.258037	.
30	.254809	.
40	.253338	.

50	.258467	.
60	.256798	.
70	.253252	.
80	.2579	.
90	.257371	.
100	.255078	.
110	.255872	.
120	.253553	.
130	.258467	.
140	.256771	.
150	.25815	.
160	.26053	.
170	.258156	.
180	.254505	.
190	.258956	.
200	.261025	.
210	.255869	.
220	.257042	.
230	.255696	.
240	.257031	.
250	.257627	.
260	.253751	.
270	.255936	.
280	.252861	.
290	.254893	.
300	.253456	.
310	.253051	.

END AT 8054

### §3 快速富里叶分解

#### 一、方法概要

快速富里叶变换已经成为数字信号处理和线性系统分析的有力工具。这里介绍的只是单纯的快速富里叶变换，实际上在它的使用中，一定要和一定的“窗函数”联合起来使用才行。

下面，我们概要地叙述一下算法，首先从离散富里叶变换谈起，离散富里叶变换简化写成DFT。

根据连续富里叶变换，我们知道，如果 $x(t)$ 是周期函数，周期为 $T$ ，则能写成：

$$x(t) = a_0 + 2 \sum_{k=1}^{\infty} \left( a_k \cos \frac{2\pi kt}{T} + b_k \sin \frac{2\pi kt}{T} \right) \quad (13-3-1)$$

$$\text{其中 } a_k = \frac{1}{T} \int_0^T x(t) \cos \frac{2\pi kt}{T} dt$$

$$b_k = \frac{1}{T} \int_0^T x(t) \sin \frac{2\pi kt}{T} dt \quad (13-3-2)$$

若我们使用复数符号, 定义

$$X_k = a_k - ib_k \quad (13-3-3)$$

并使用  $e^{-i(2\pi kt/T)} = \cos \frac{2\pi kt}{T} - i \sin \frac{2\pi kt}{T}$ , 就能把 (13-3-2) 中的两个式子合并成一个式子为:

$$X_k \stackrel{k \geq 0}{=} \frac{1}{T} \int_0^T x(t) e^{-i(2\pi kt/T)} dt \quad (13-3-4)$$

现在考虑, 若连续时间函数  $x(t)$  不知道, 只有等间隔的采样函数可用, 情况如何呢?

假定这个采样函数是一个离散的级数  $\{x_r\}$ ,  $r=0, 1, 2, \dots, (N-1)$ , 其中  $t=r\Delta$ ,  $\Delta=T/N$  为采样间隔, 如图 (13-3-1) 所示:

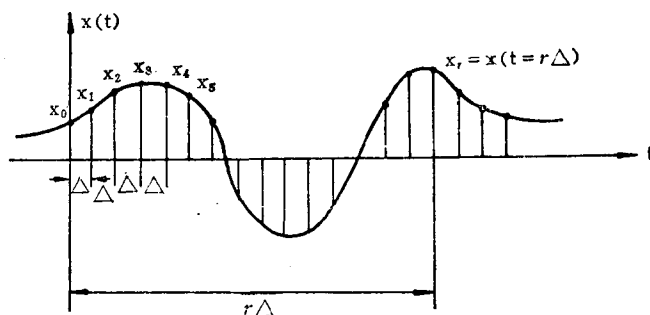


图 13-3-1 在定间隔上对连续函数取样

在这种情况下, 式 (13-3-4) 中的积分可以近似地用累加代替, 即:

$$X_k = \frac{1}{T} \sum_{r=0}^{N-1} x_r e^{-i(2\pi k/T)(r\Delta)} \quad (13-3-5)$$

这相当于图 (13-3-2) 所示曲线下的面积为所有阴影片条的总和

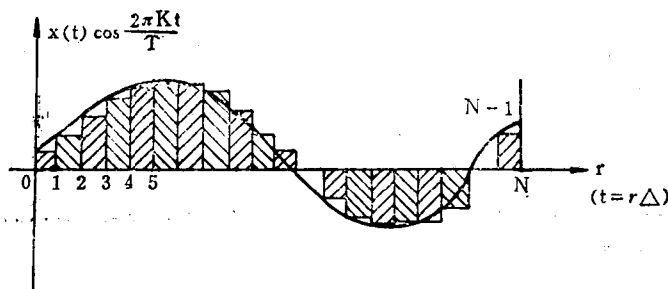


图 13-3-2 根据离散级数而不是连续函数计算富里叶系数的近似

将  $T=\Delta N$  代入 (13-3-5) 式中, 则:

$$X_k = \frac{1}{N} \sum_{r=0}^{N-1} x_r e^{-i(2\pi kr/N)} \quad k=0, \dots, N-1 \quad (13-3-6)$$

式(13-3-6)可看作是计算(13-3-1)式的富里叶级数的系数的近似公式。

虽然(13-3-6)式并不能提供足够的信息,以允许求得连续时间函数 $x(t)$ ,但是最为重要的是它允许精确地回到函数 $\{x_r\}$ 的所有离散值。

即有(13-3-6)式的逆变换为:

$$x_r = \sum_{k=0}^{N-1} X_k e^{i(2\pi k r/N)} \quad (13-3-7)$$

式(13-3-7)可以给出函数 $\{x_r\}$ 中的任何一个典型值 $x_r$ ,  $r=0,1,2,\dots,N-1$ 。

证明:为避免搞混后面的项,将(13-3-6)式中的 $r$ 取为 $S$ ,即 $r=S$ ,则(13-3-7)式的右边为:

$$\begin{aligned} \sum_{k=0}^{N-1} X_k e^{i(2\pi k r/N)} &= \sum_{k=0}^{N-1} \left\{ \frac{1}{N} \sum_{s=0}^{N-1} x_s e^{-i(2\pi k s/N)} \right\} e^{i(2\pi k r/N)} \\ &= \sum_{k=0}^{N-1} \sum_{s=0}^{N-1} \frac{1}{N} e^{-i(2\pi k/N)(s-r)} \end{aligned}$$

改变累加次序。使上式为

$$= \sum_{s=0}^{N-1} \left\{ \sum_{k=0}^{N-1} e^{-i(2\pi k/N)(s-r)} \right\} \frac{1}{N} x_s$$

因为 $K, S, r$ 和 $N$ 都是整数,除非 $S=r$ 则其余各种情况指数函数的和都等于零,即:

$$\left\{ \sum_{k=0}^{N-1} e^{-i(2\pi k/N)(s-r)} \right\} = 0 \quad s \neq r$$

$$\text{而: } \left\{ \sum_{k=0}^{N-1} e^{-i(2\pi k/N)(s-r)} \right\} = N \quad s=r$$

∴上式为

$$\sum_{s=0}^{N-1} \left\{ \sum_{k=0}^{N-1} e^{-i(2\pi k/N)(s-r)} \right\} \frac{1}{N} x_s = x_r$$

∴上式右边等于左边,得证。

∴我们得到一对离散富里叶变换DFT和IDFT。

$$\text{DFT: } X_k = \frac{1}{N} \sum_{r=0}^{N-1} x_r e^{-i(2\pi k r/N)} \quad k=0,1,\dots,N-1 \quad (13-3-8)$$

$$\text{IDFT: } x_r = \sum_{k=0}^{N-1} X_k e^{i(2\pi k r/N)} \quad r=0,1,2,\dots,N-1 \quad (13-3-9)$$

为保供变换对(13-3-8)和(13-3-9)的对称性,把富里叶分量 $X_k$ 的范围限于 $K=0$ 到 $N-1$ , (同频率 $\omega_k=2\pi k/N=2\pi k/N\Delta$ 的简谐分量相对应)。

虽然我们是从研究连续函数的富里叶变换的性质来引出DFT,但是,重要的是,离散富里叶变换对(13-3-8)和(13-3-9)是精确的,严密的,而没有任何的近似。

式(13-3-8)确定函数 $\{x_r\}$   $r=0,1,\dots,N-1$ 的DFT。但是如果我们试图计算 $K>N-1$ 的 $X_k$ 值情况又怎样呢?

令  $K=N+L$

$$\begin{aligned} \text{则: } X_{N+L} &= \frac{1}{N} \sum_{r=0}^{N-1} x_r e^{-i(2\pi r/N)(N+L)} \\ &= \frac{1}{N} \sum_{r=0}^{N-1} x_r e^{-i(2\pi L/N)} e^{-i2\pi r} \end{aligned}$$

∴ 不管  $r$  为任何值  $e^{-i2\pi r} \equiv 1$

(13-3-10)

∴  $X_{N+L} = X_L$

∴ 当  $K > (N-1)$  时系数  $X_K$  刚好重复自己, 所以如果我们把  $|x_K|$  的大小沿频率轴  $\omega_K = 2\pi k/N\Delta$  画出来, 则曲线如图 (13-3-3) 所示那样, 周期地重复自己, 而且根据 (13-3-8)

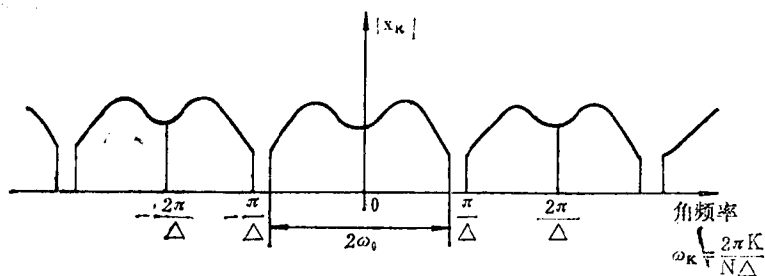


图 13-3-3 用DFT计算富里叶系数的周期性

式也可以容易看到, 倘使  $\{x_r\}$  的级数中的项是实数, 那么

$$X_{-L} = X_L^* \quad (X_L \text{ 的共轭复数})$$

(13-3-11)

∴  $|X_{-L}| = |X_L|$

如图 (13-3-4) 中所示那样, 曲线图对零频位置是对称的, 其中只有曲线中的一部分 (即  $|\omega| \leq \frac{\pi}{\Delta}$ ) 才是真的富里叶系数, 而较高频率指出的都是假的富里叶系数, 它是  $|\omega| \leq \frac{\pi}{\Delta}$  那一部分的周期性重复。所以用DFT计算的系数  $X_K$  只对

$$\omega = \frac{2\pi k}{N\Delta} \leq \frac{\pi}{\Delta} \quad (\text{rad/s}) \quad k \text{ 在 } 0, 1, 2, \dots, \frac{N}{2} \text{ 范围内的频率才是正确的富里叶系数。正因为如此, 如果在原始信号里有 } \pi/\Delta \text{ 以上的频率, 将会产生称之为“混淆”的曲线畸变, 如图 (13-3-4) 所示, 高频分量对 } \{x_r\} \text{ 函数产生影响, 它使得DFT计算的低于 } \pi/\Delta \text{ 频率的富里叶系数发生畸变。如果 } \omega_0 \text{ 是 } x(t) \text{ 中存在的最高频率分量, 那么保证取样的间隔 } \Delta \text{ 足够小, 即, } \frac{\pi}{\Delta} > \omega_0, \text{ 就能够避免混淆, 或者, 如果 } f_0 = \omega_0/2\pi, \text{ 则保证 } \frac{1}{2\Delta} > f_0.$$

频率  $\frac{1}{2\Delta}$  (赫兹) 称之为奈奎斯特频率 (有时称为折叠频率)。

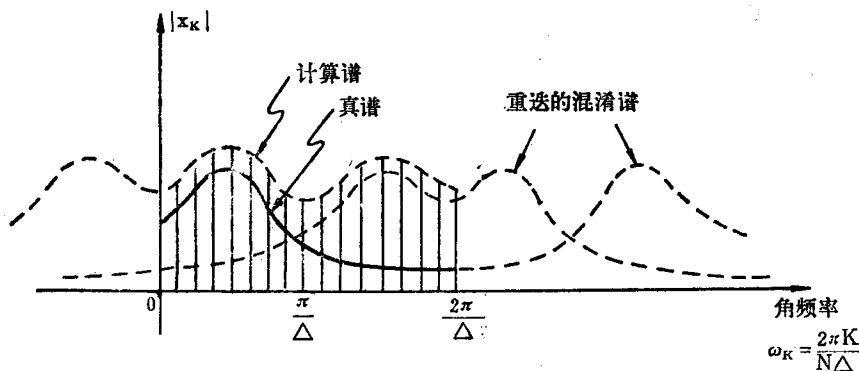


图 13-3-4 信号带宽超过  $\pi/\Delta$  rad/s 时的混淆畸变

在分析实际数据时混淆现象最为重要。为了包括连续信号的整个频率范围, 取样频率  $1/2\Delta$  必须足够高。换句话说, 有时, 为消去比  $1/2\Delta$  高的频率分量, 必须先经过滤波才行。

下面再谈谈快速富里叶变换。快速富里叶变换简称为FFT, 它只是DFT的一种算法。

没有什么新的概念。

FFT是将整个序列 $\{x_r\}$ 分割成若干较短的序列来工作,代替计算原始序列的DFT。即只要算出较短序列的DFT,然后FFT就以巧妙的方法把它们并在一起,得出整个的 $\{x_r\}$ 的DFT。这看来象很复杂,实际上却是惊人的简单。

假定 $\{x_r\}, r=0, 1, 2, \dots, N-1$ 是一个图(13-3-5)(a)所示的序列,其中 $N$ 是偶数,把它分成如图(13-3-5)(b)所示的两个较短的序列 $\{y_r\}$ 和 $\{z_r\}$ ,其中

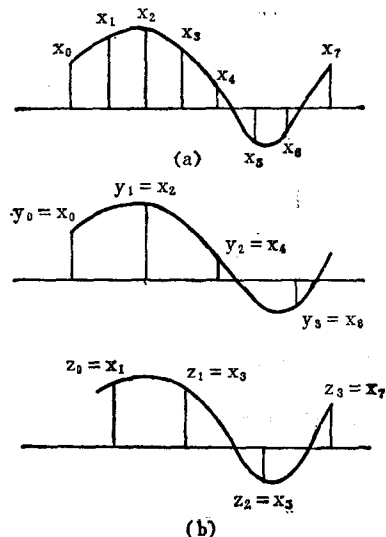


图 13-3-5

$$\left. \begin{aligned} y_r &= x_{2r} \\ z_r &= x_{2r+1} \end{aligned} \right\} r=0, 1, 2, \dots, \left(\frac{N}{2}-1\right) \quad (13-3-12)$$

这两个短序列的DFT分别是 $Y_k$ 和 $Z_k$ , 根据(13-3-8)式

$$Y_k = \frac{1}{(N/2)} \sum_{r=0}^{N/2-1} y_r e^{-j \frac{2\pi k r}{(N/2)}} \quad k=0, 1, 2, \dots, (N/2-1)$$

$$Z_k = \frac{1}{(N/2)} \sum_{r=0}^{N/2-1} z_r e^{-j \frac{2\pi k r}{(N/2)}} \quad (13-3-13)$$

而

$$X_k = \frac{1}{N} \sum_{r=0}^{N-1} x_r e^{-j \left(\frac{2\pi k r}{N}\right)}$$

$$= \frac{1}{N} \left\{ \sum_{r=0}^{N/2-1} x_{2r} e^{-j \left(\frac{2\pi (2r) k}{N}\right)} + \sum_{r=0}^{N/2-1} x_{2r+1} e^{-j \left(\frac{2\pi (2r+1) k}{N}\right)} \right\}$$

将(13-3-12)式代入得:

$$X_k = \frac{1}{N} \left\{ \sum_{r=0}^{N/2-1} y_r e^{-j \left(\frac{2\pi k r}{N/2}\right)} + \left( \sum_{r=0}^{N/2-1} z_r e^{-j \left(\frac{2\pi k r}{N/2}\right)} \right) \cdot e^{-j \frac{2\pi k}{N}} \right\}$$

将上式与(13-3-13)式比较得出:

$$X_k = \frac{1}{2} \left\{ Y_k + e^{-j \left(\frac{2\pi k}{N}\right)} \cdot Z_k \right\} \quad k=0, 1, \dots, (N/2-1) \quad (13-3-14)$$

因此,按照(13-3-14)式的算法,原始序列的DFT能直接根据两个半序列 $y_r$ 和 $z_r$ 的DFT求得,这式子是FFT算法的中心。如果序列 $\{x_r\}$ 中的原始数量 $N$ 是2的乘幂,则半序列 $\{y_r\}$ 和 $\{z_r\}$ 自己又可以分隔成1/4序列,等等,直到最后每个子序列只有一项为止。

算法(13-3-14)只适用于在0到 $N/2-1$ 之间的 $K$ 值,即它只适用于 $\{x_k\}$ 序列的一半系数。而我们须要 $K$ 从0到 $(N-1)$ 的 $X_k$ 。因此把 $N/2$ 到 $(N-1)$ 的一半系数加到算法中去,结果如式(13-3-10)所示的那样,即 $Y_k$ 和 $Z_k$ 对 $K$ 是周期性的,并以 $N/2$ 的周期重复它们自己。所以:

$$Y_{k-N/2}=Y_k, Z_{k-N/2}=Z_k$$

∴ 由 $Y_k$ 和 $Z_k$ 计算 $X_k$ 的全部系数的计算方法为:

$$X_k = \frac{1}{2} \left\{ Y_k + e^{-i\left(\frac{2\pi k}{N}\right)} \cdot Z_k \right\} \quad k=0, 1, \dots, (N/2-1) \quad (13-3-15)$$

$$X_k = \frac{1}{2} \left\{ Y_{k-N/2} + e^{-i\left(\frac{2\pi k}{N}\right)} \cdot Z_{k-N/2} \right\} \quad k = \frac{N}{2}, \left(\frac{N}{2}+1\right), \dots, (N-1) \quad (13-3-16)$$

或者,如果我们只允许 $k$ 从0到 $N/2$ ,则另一种等效算法是:

$$X_k = \frac{1}{2} \left\{ Y_k + e^{-i\left(\frac{2\pi k}{N}\right)} \cdot Z_k \right\} \quad k=0, 1, \dots, \left(\frac{N}{2}-1\right)$$

$$X_{k+\frac{N}{2}} = \frac{1}{2} \left\{ Y_k + e^{-i\left(\frac{2\pi(k+\frac{N}{2})}{N}\right)} \cdot Z_k \right\} \quad k=0, 1, \dots, \left(\frac{N}{2}-1\right) \quad (13-3-17)$$

利用 $e^{-i\pi} = -1$ ,上式简化成:

$$\begin{aligned} X_k &= \frac{1}{2} \left\{ Y_k + e^{-i(2\pi k/N)} \cdot Z_k \right\} \quad k=0, 1, 2, \dots, \left(\frac{N}{2}-1\right) \\ X_{k+\frac{N}{2}} &= \frac{1}{2} \left\{ Y_k - e^{-i(2\pi k/N)} \cdot Z_k \right\} \end{aligned} \quad (13-3-18)$$

最后,如果我们定义新的复变量 $\omega$ ,

$$\text{使 } \omega = e^{-i(2\pi/N)} \quad (13-3-19)$$

就能得到称之为“蝶形线路”的计算公式:

$$\begin{aligned} X_k &= \frac{1}{2} \{ Y_k + \omega^k Z_k \} \\ &\quad k=0, 1, 2, \dots, (N/2-1) \\ X_{k+N/2} &= \frac{1}{2} \{ Y_k - \omega^k Z_k \} \end{aligned} \quad (13-3-20)$$

式(13-3-20)所表明的算法几乎在所有的FFT计算机程序中都有。

用一个例子来说明上面所讲的计算步骤。假定 $\{X_r\}$ 只有四项,它能分隔成图(13-3-6)所示的子序列,直到1/4序列中每个只有一项。考虑DFT的基本定义(13-3-8)式,显然,如果序列只有一项,

$$\text{则 } X_k = \frac{1}{N} \sum_{r=0}^{N-1} X_r e^{-i\frac{2\pi k r}{N}} = X \quad N=1, r=0, k=0$$



所以单项序列的 DFT 就等于单项自己。因此, 我们很快就知道“1/4”序列的 DFT, 并且只要用二步合并它们, 就可求出图 (13-3-6) 所示下半部分的 DFT, 用图中的符号, 则四个“1/4”序列的 DFT 为:

$$\{T_k\} = x_0, \{U_k\} = x_2, \{V_k\} = x_1, \{W_k\} = x_3$$

根据式 (13-3-20),  $N/2=1$ , 则:

$$Y_0 = \frac{1}{2}\{x_0 + x_2\}, \quad Z_0 = \frac{1}{2}\{x_1 + x_3\}$$

$$\omega = e^{-i(2\pi/N)} = e^{-i\pi} = -1 \quad (13-3-21)$$

$$Y_1 = \frac{1}{2}\{x_0 - x_2\}, \quad Z_1 = \frac{1}{2}\{x_1 - x_3\}$$

第二次应用 (13-3-20) 式,  $N/2=2$ ,

$$\omega = e^{-i2\pi/N} = e^{-i\pi/2} = -i$$

$$\therefore X_0 = \frac{1}{4}\{x_0 + x_2 + x_1 + x_3\}$$

$$X_1 = \frac{1}{4}\{x_0 - x_2 - i(x_1 - x_3)\}$$

$$X_2 = \frac{1}{4}\{x_0 + x_2 - (x_1 + x_3)\} \quad (13-3-22)$$

$$X_3 = \frac{1}{4}\{x_0 - x_2 + i(x_1 - x_3)\}$$

由 (13-3-8) 式给出的  $\{x_r\}$  的定义, 很容易核对它正好与以上结果相同。

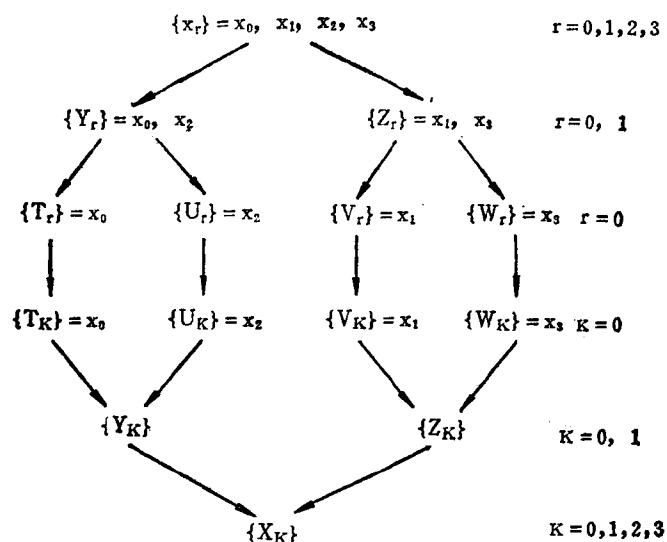


图 13-3-6 四项序列  $\{X_r\}$  的 FFT 计算步骤

“蝶形线路图”——图 (13-3-7) 表示了 FFT 计算步骤。其上面两个“蝶形线路”表示 (13-3-21) 式, 下面两个重迭的蝶形线路表示 (13-3-22) 式。

扩展到  $N=2^m$  项的情况见图 (13-3-8) 所示的步骤, 计算的次序是从一排蝶形线路相继进行到下一排。对  $m$  排来说, 把数据构成  $2^m$  个集群, 每个蝶形线路有着  $K = \frac{1}{2} \cdot 2^m = 2^{m-1}$  项

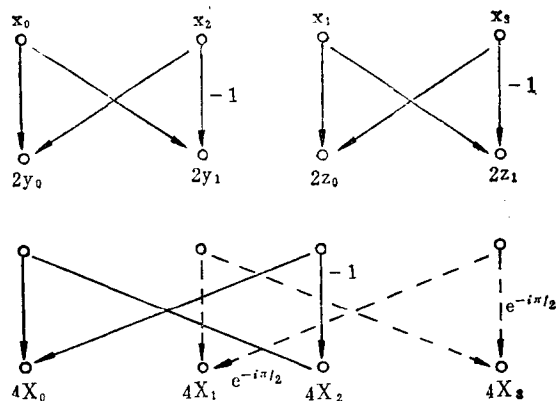


图 13-3-7 FFT 算法中的连续步骤

的密度，乘子是从  $W_m = e^{-i\pi/K}$  开始，以上升乘幂出现，直到  $W_m^{(K-1)}$ 。

图 (13-3-9) 是进行上面计算的程序流程图。

作为图 (13-3-9) 所示的计算程序之前，还有两件事要做：

(1) 要用  $N$  除每一个原始数据，代替蝶形线路计算的每个阶段上用 2 除。

所以，同原始序列  $x_0, x_1, x_2, x_3$  相对应，有  $a_1 = \frac{x_0}{4}, a_2 = \frac{x_1}{4}, a_3 = \frac{x_2}{4}, a_4 = \frac{x_3}{4}$ 。

对应原始序列  $x_0, x_1, \dots, x_7$  相对应，有  $a_1 = \frac{x_0}{8}, a_2 = \frac{x_1}{8}, \dots, a_8 = \frac{x_7}{8}, \dots$ 。

(2) 按照图 (13-3-6) 在把  $N$  项初始序列分隔成  $N$  个单项序列的过程中，改变了初始序列的次序。

如初始次序是：1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16。那么分隔后的次序是：1, 9, 5, 13, 3, 11, 7, 15, 2, 10, 6, 14, 4, 12, 8, 16。

写出相邻两项数之差，为：

8, -4, 8, -10, 8, -4, 8, -13, 8, -4, 8, -10, 8, -4, 8。

显然，项数差 8 有规律地出现，如果我们仔细看一下，就能见到，若  $j$  是序列中的任何一个项数，那么如果  $j \leq 8$ ，下一个项数总是  $(j+8)$ 。同样， $8 < j \leq 12$ ，则下一项数总是  $(j-4)$ ；如果  $12 < j \leq 14$ ，则  $j$  后总是  $(j-10)$ ，如果  $14 < j \leq 15$ ，则  $j$  后是  $(j-13)$ 。

在图 (13-3-10) 中，我们表示分隔 32 项序列的结果，能从这结果中推论出计算重新排列中上升项的项数的一般“算法”。如果序列长  $N=2^n$ ，那么如果  $j$  是重新排列中的任意一项的项数，则下一项的项数将是：

$$j + \frac{N}{2} \quad \text{如果} \quad j \leq \frac{N}{2}$$

$$j + \left(\frac{N}{4} - \frac{N}{2}\right) \quad \text{如果} \quad \frac{N}{2} < j \leq \frac{N}{2} + \frac{N}{4}$$

$$j + \left(\frac{N}{8} - \frac{N}{4} - \frac{N}{2}\right) \quad \text{如果} \quad \frac{N}{2} + \frac{N}{4} < j \leq \frac{N}{2} + \frac{N}{4} + \frac{N}{8}$$

.....

$$j + (1 - 2 - 4 - \dots - \frac{N}{2}) \quad \text{如果} \quad (\frac{N}{2} + \dots + 4 + 2) < j \leq (\frac{N}{2} + \dots + 4 + 2 + 1) \\ = (N-1)$$

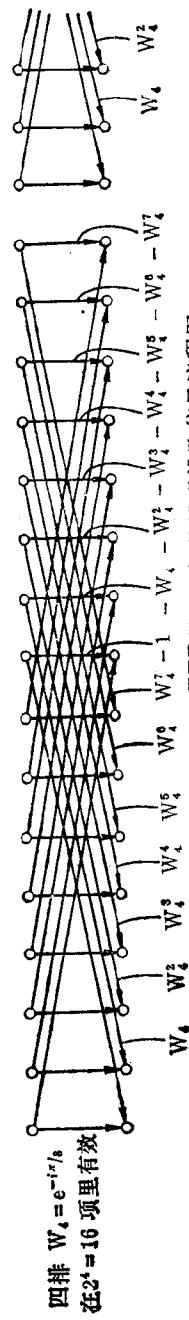
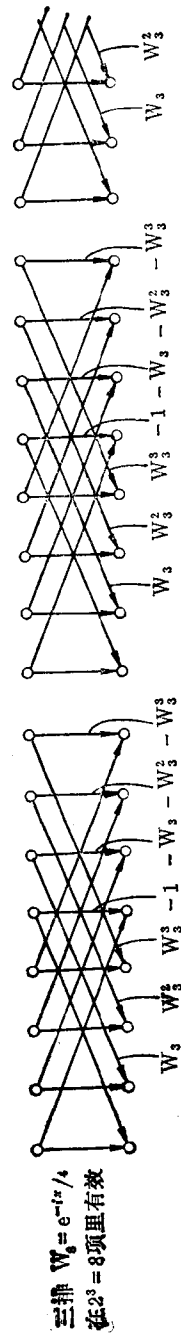
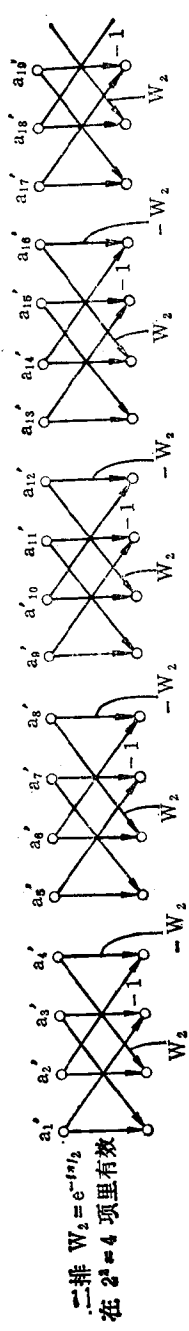
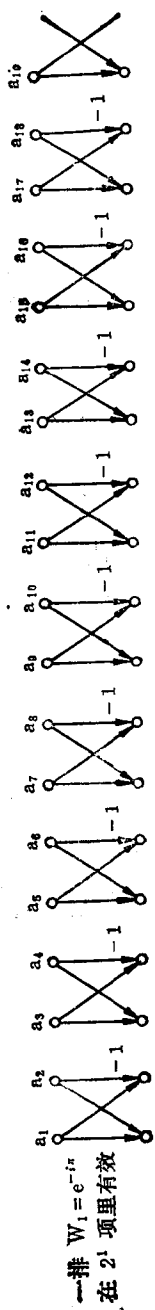


图 13-3-8 说明FFT程序的逻辑的蝶形线路信号流程图

原始序列

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32
1	5	9	13	17	21	25	29	3	7	11	15	19	23	27	31	2	6	10	14	18	22	26	30	4	8	12	16	20	24	28	32
1	9	17	25	5	13	21	29	3	11	19	27	7	15	23	31	2	10	18	26	6	14	22	30	4	12	20	28	8	16	24	32
1	17	9	25	5	21	13	29	3	19	11	27	7	23	15	31	2	18	10	26	6	22	14	30	4	20	12	28	8	24	16	32

分成  $\frac{1}{2}$  序列

分成  $\frac{1}{4}$  序列

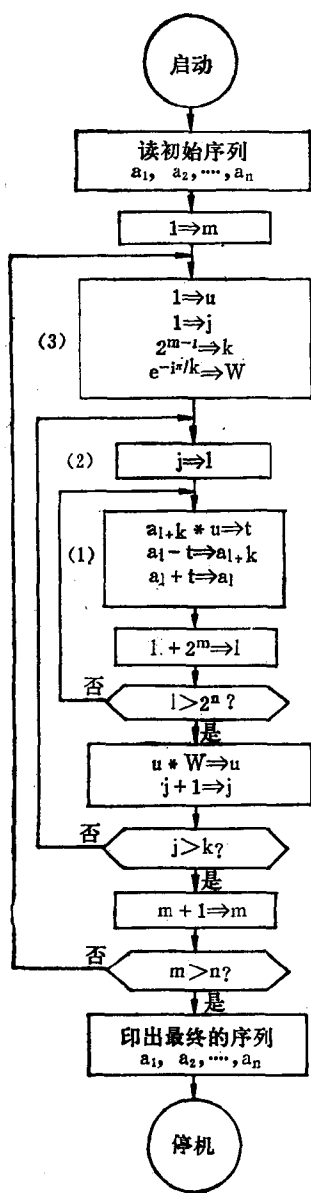
分成  $\frac{1}{8}$  序列

分成  $\frac{1}{16}$  序列

在完全被分隔后的序列中相邻两项数之差表

$+\frac{N}{2}$	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
$(\frac{N}{4}-\frac{N}{2})$	-8		-8		-8		-8		-8		-8		-8		-8		-8		-8		-8		-8		-8		-8		-8		-8
$(\frac{N}{8}-\frac{N}{4}-\frac{N}{2})$		-20									-20										-20										-20
$(\frac{N}{16}-\frac{N}{8}-\frac{N}{4}-\frac{N}{2})$												-26													-26						
$(\frac{N}{32}-\frac{N}{16}-\frac{N}{8}-\frac{N}{4}-\frac{N}{2})$																-29															

图 13-3-10 32项重排序列的相邻两项之差的表



### 注 解

用  $N$  除每一项之后, 按二进制颠倒次序写出的序列

$m$ ——第  $n$  排蝶形线路

$K$ ——每排的“蝶形”的宽度

$j$ ——在一排中重复进行  $a_l \pm a_{l+k} \cdot \omega^j$

$t, u$  是工作单元

环 (1) 是一排中从一块蝶形线路到另一块蝶形线路的计算。

环 (2) 是一排内, 一块蝶形线路当中, 从一个元素到下一个元素的计算。

环 (3) 从一排到另一排的循环计算。

图 13-3-10 简单FFT的程序流程图

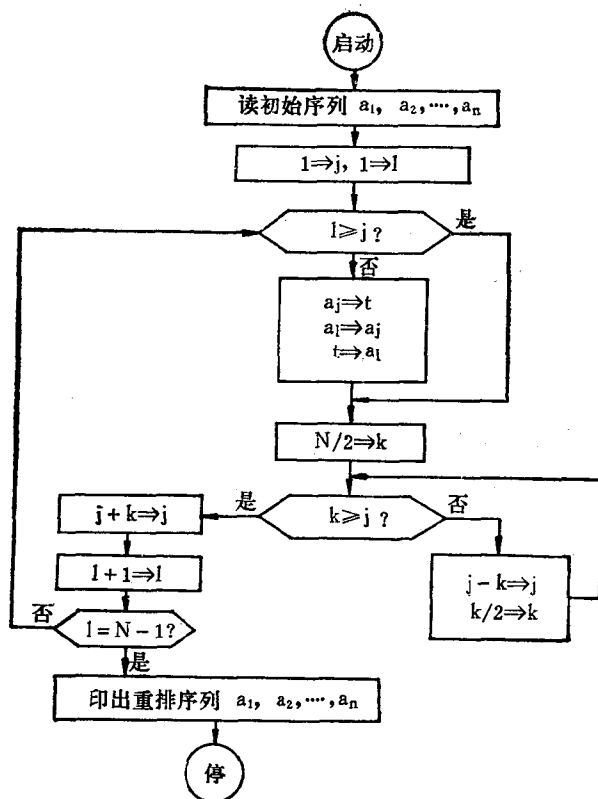


图 13-3-11 给出了整个计算的流程图

以上所述的称为基数为 2 的 FFT 算法，原始序列长度  $N$  必须是 2 的乘幂，所以原始序列能分隔成两个等长的  $1/2$  序列等等。当  $N$  不是 2 的乘幂时，而有别的公因子，则要逐步修改蝶形公式。相同的基本概念是适用的。

但  $N=2^n$  的算法简单，通过加零来人为地增长序列或是有意识地选取  $N=2^n$  的值，对于大多数谱分析应用是足够的。

## 二、程序说明

1. 从语句标号 8002 到语句标号 8090 是 FFT 子程序，从 8092 到 8174 是使用 FFT 的假想主程序。

2. 子程序使用数组 A，加工的是复数序列。其实部是离散时间的数值  $x_0, x_1, \dots, x_{N-1}$  (实数)，而其虚部是零，即  $A(2 * I)$  是实部， $A(2 * I + 1)$  存的是零，从子程序转出时 A 数组也是复数序列  $X_0, X_1, \dots, X_{N-1}$ 。 $A(2 * I)$  是实部， $A(2 * I + 1)$  是虚部。

3. 使用工作单元：

$N$ ——数据个数，其一定是 2 的幂方数，设其为  $s$ —— $N=2^s$ ，

$T$ —— $\begin{cases} = 1 & \text{逆变换} \\ = -1 & \text{正变换} \end{cases}$

其他工作单元有  $T_1, T_2, U_1, U_2, W_1, W_2, M, P_1, K, L, L_1$ ，循环变量有  $I, J, L$  等。

4. 从子程序转出，原来的信息破坏。

## 三、程序使用

1. 由 BASIC 引入本程序，可以作为子程序使用，只要预先准备好 A 数组的数据以

及  $N$ ,  $S$ ,  $T$  的值, 就可以用 GOSUB 8002 转入, 转出时结果在  $A$  数组中。

2. 就本程序所准备的假想主程序来说, 只要用键盘命令 RUN 启动后, 机器自动印出: "INPUT NOM. OF COMPLEX DATA  $N$  AND  $S$ ", 询问原始数据序列的长度  $N$  和  $S$  值 ( $N=2^8$ )。

3. 当使用者由键盘回答  $N$  和  $S$  值后, 机器立即印出 "SERIES" 和 "CONVERTION," 以后 "H(F)", "PSD" 等英文字样并有相应的数据印出, 它们分别是原始数据序列、富里叶变换后的数据序列、幅频特性数据和功率谱数据。

#### 四、试题

$$\{x_r\} = \{\cos 2\pi f r\} \quad f=2$$
$$r=0, 1, 2, \dots, N$$
$$N=32 \quad \Delta f=1/32$$

取  $\Delta=1$  秒

#### 五、程序清单及运行结果

##### 程序清单

```
8000 GOTO 8092
8002 LET N1=N/2
8004 LET N2=N-1
8006 LET J=1
8008 FOR L=1 TO N2
8010 IF L=>J THEN GOTO 8024
8012 LET T1=A(2*J)
8014 LET T2=A(2*J+1)
8016 LET A(2*J)=A(2*L)
8018 LET A(2*J+1)=A(2*L+1)
8020 LET A(2*L)=T1
8022 LET A(2*L+1)=T2
8024 LET K=N1
8026 IF K=>J THEN GOTO 8034
8028 LET J=J-K
8030 LET K=K/2
8032 GOTO 8026
8034 LET J=J+K
8036 NEXT L
8038 LET P1=3.14159
8040 FOR I=1 TO S
8042 LET U1=1
8044 LET U2=0
8046 LET M=1
8048 FOR L=1 TO I
8050 LET M=M*2
8052 NEXT L
```

```

8054 LET K=M/2
8056 LET W1=COS(P1/K)
8058 LET W2=T * SIN(P1/K)
8060 FOR J=0 TO K
8062   FOR L=J TO N STEP M
8064     LET L1=L+K
8066     LET T1=A(2 * L1) * U1-A(2 * L1+1) * U2
8068     LET T2=A(2 * L1) * U2+A(2 * L1+1) * U1
8070     LET A(2 * L1)=A(2 * L)-T1
8072     LET A(2 * L1+1)=A(2 * L+1)-T2
8074     LET A(2 * L)=A(2 * L)+T1
8076     LET A(2 * L+1)=A(2 * L+1)+T2
8078   NEXT L
8080   LET Z=U1 * W1-U2 * W2
8082   LET U2=U1 * W2+U2 * W1
8084   LET U1=Z
8086 NEXT J
8088 NEXT I
8090 RETURN
8092 PRINT
8094 PRINT "INPUT NOM. OF COMPLEX DATA N AND S"
8096 INPUT N, S
8098 DIM A(2 * N+2)
8100 PRINT
8102 LET X=2 * 3.14159 * 2/N
8104 PRINT "SERITES"
8106 PRINT
8108 FOR I=1 TO N
8110   LET A(2 * I)=COS((I-1) * X)
8112   LET A(2 * I+1)=0
8114   PRINT A(2 * I),
8116   LET A(2 * I)=A(2 * I)/N
8118 NEXT I
8120 PRINT
8122 LET T=-1
8124 GOSUB 8002
8126 PRINT "CONVERSION;"
8128 PRINT
8130 FOR I=1 TO N
8132   PRINT A(2 * I); "+I"; A(2 * I+1),

```



```

8134 NEXT I
8136 PRINT "H(F):"
8138 PRINT
8140 FOR I=1 TO N
8142 LET W=SQR(A(2*I)*A(2*I)+A(2*I+1)*A(2*I+1))
8144 PRINT W,
8146 NEXT I
8148 PRINT
8150 PRINT "PSD:"
8152 PRINT
8154 FOR I=1 TO N
8156 LET W=(A(2*I)*A(2*I)+A(2*I+1)*A(2*I+1))
8158 IF I=1 THEN GOTO 8162
8160 LET W=2*W
8162 PRINT W,
8164 NEXT I
8166 END

```

运行结果

RUN

INPUT NOM. OF COMPLEX DATA N AND S

? 32? 5

SERITES

.999999	.92388	.707108	.382686
3.21865E-6	-.38268	-.707104	-.923877
-.999999	-.923882	-.707112	-.382691
-8.22544E-6	.382675	.707101	.923875
.999999	.923884	.707115	.382694
1.28746E-5	-.382672	-.707098	-.923874
-.999999	-.923885	-.707118	-.382698
-1.57356E-5	.38267	.707094	.923872

CONVERSION,

-1.36718E-6	+ I 0	-1.75026E-6	+ I 1.28246E-7
.499999	+ I -3.75509E-6	1.02425E-6	+ I 1.3441E-7
4.59237E-7	+ I -3.68003E-8	2.26203E-7	+ I -5.26944E-8
-1.20649E-6	+ I -2.51762E-7	1.47389E-7	+ I -4.53756E-8
1.41561E-7	+ I -8.56817E-8	1.81478E-7	+ I -2.55724E-8
-3.2116E-7	+ I -3.16465E-7	1.32434E-7	+ I 1.67624E-8
3.25002E-8	+ I 5.26049E-8	1.49301E-7	+ I 1.41985E-8
-1.19209E-7	+ I -1.47521E-6	-8.0986E-8	+ I -4.16031E-8
1.00583E-7	+ I 0	-8.09853E-8	+ I 4.16063E-8

2.98023E-8	+ I -4.17234E-7	1.49302E-7	+ I -1.41994E-8
3.25009E-8	+ I -5.26068E-8	1.32434E-7	+ I -1.67621E-8
2.82622E-7	+ I -7.31712E-7	1.81476E-7	+ I 2.55758E-8
1.41561E-7	+ I 8.56817E-8	1.47387E-7	+ I 4.53775E-8
3.2116E-7	+ I -5.18004E-7	2.26201E-7	+ I 5.26967E-8
4.59238E-7	+ I 3.68021E-8	1.02425E-6	+ I -1.34399E-7
.5	+ I 7.46548E-6	-1.75027E-6	+ I -1.28267E-7

H(F):

1.36718E-6	1.75495E-6	.499999	1.03304E-6
4.6071E-7	2.32259E-7	1.23248E-6	1.54216E-7
1.65472E-7	1.8327E-7	4.50881E-7	1.3349E-7
6.18348E-8	1.49974E-7	1.48002E-6	9.10469E-8
1.00583E-7	9.10477E-8	4.18297E-7	1.49976E-7
6.18367E-8	1.3349E-7	7.84396E-7	1.8327E-7
1.65471E-7	1.54214E-7	6.09485E-7	2.32258E-7
4.6071E-7	1.03303E-6	.5	1.75496E-6

PSD:

1.86919E-12	6.15973E-12	.499999	2.13432E-12
4.24507E-13	1.07889E-13	3.03802E-12	4.7565E-14
5.47619E-14	6.71761E-14	4.06587E-13	4.56393E-14
7.64708E-15	4.49847E-14	4.38093E-12	1.65791E-14
2.02338E-14	1.65794E-14	3.49945E-13	4.49855E-14
7.64757E-15	3.56394E-14	1.23055E-12	6.71755E-14
5.47616E-14	4.75639E-14	7.42943E-13	1.07888E-13
4.24508E-13	2.1343E-12	.500001	6.15978E-12

END AT 8166

## §4 用快速富里叶变换 (FFT) 求相关函数

### 一、计算方法概要

计算相关函数的间接方法, 就是使用快速富里叶变换 (FFT) 求相关函数。间接方法的最大优点是占用计算机的机时要少, 原始数据越多, 相关序列越长, 这个优点越显著。例如, 用直接方法计算  $m$  个滞后时间间隔的相关函数, 需要  $2Nm$  次实数的乘加运算; 而用 FFT, 若  $N=2^p$ , 则其需要实数的乘加运算为  $8Np$  次, 因此两种方法的速度比是,

$$SR = \frac{2mN}{8Np} = \frac{m}{4p}$$

若取  $N=8192$ ,  $p=13$ ,  $m=400$ ,

$$\text{则 } SR = \frac{400}{4 \times 13} = 7.7$$

即用 FFT 方法要比用直接法快将近八倍。

使用间接方法要注意的是,这种方法一般不能得到通常的相关函数,而得到的是“循环相关函数”。它是由下式定义的:

$$R_x^c(\gamma) = \frac{(N-\gamma)}{N} \left[ R_x(\gamma) + R_x(N-1-\gamma) \right] \quad (13-4-1)$$

$\gamma = 0, 1, 2, \dots, N-1$

其中 $R_x(\gamma)$ 是由直接法求出的相关函数。

并且有:

$$R_x(N-1-\gamma) = R_x(\gamma) \quad (13-4-2)$$

$\gamma = 0, 1, 2, \dots, N-1$

在图(13-4-1)中具体描述了式(13-4-1)的两部分。

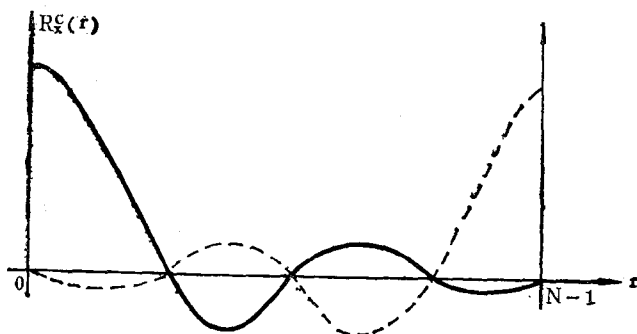


图 13-4-1

实际上对于衰减很快的相关函数来说,滞后 $\gamma$ 比较小(例如小于 $N$ 的20%),可以不考虑“循环相关”的影响。但是不管什么情况,只要在原始数据序列中添上一些零点,就可以避免这个问题。添加零点对相关函数的作用,就是把循环相关的两部分分开。特别是,如果在原始 $N$ 个数据值的序列中再添加 $N$ 个零,则两部分可以完全分开,见图(13-4-2)

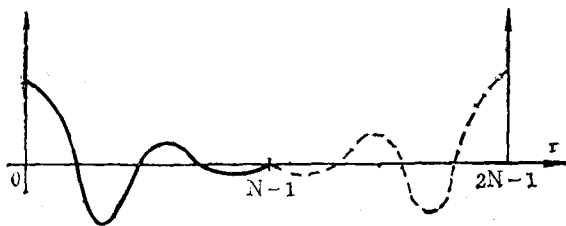


图 13-4-2

用FFT作相关函数的步骤如下:

1. 对 $N$ 个数据值 $x_k (k=1, \dots, N)$ 和 $y_k (k=1, 2, \dots, N)$ 分别添加 $N$ 个零得到 $2N$ 项的两个新序列。
2. 用FFT计算 $x(k)$ 和 $y(k)$ 的离散变换:

$$X(f) = \sum_{k=0}^{N-1} x(k) e^{-j2\pi f k / N}$$

$$Y(f) = \sum_{k=0}^{N-1} y(k) e^{-j2\pi f k / N}$$

3. 改变 $Y(f)$ 的虚部的符号, 得到 $Y(f)$ 的共轭复数 $Y^*(f)$ 。
4. 计算乘积 $C(f) = X(f) \cdot Y^*(f)$
5. 求 $C(f)$ 的共轭复数的FFT的逆变换:

$$A(f) = \sum_{n=0}^{N-1} \left( \frac{1}{N} C^*(n) \right) e^{-j2\pi f n / N} \quad \uparrow$$

6. 取出 $A(f)$ 的前 $N$ 点, 即为 $x(k)$ 和 $y(k)$ 的相关函数。

注: (I) 在第(2)步中, 求出的 $X(f)$ 和 $Y(f)$ 都不要乘以 $1/N$ 的比例因子。

正变(II)由第(5)步求出的 $A(f)$ 的实部即为相关函数。在这一步中是用FFT换作逆变换的, 因此取 $C^*(n)$ 。

## 二、程序说明

本程序中所使用的FFT子程序就是本章第3节中所讲的, 只是主程序稍有不同。程序中使用的工作单元见本章第3节的说明。

## 三、上机操作

1. 由BASIC引入本程序。使用者若要求改变自己的原始序列的相关函数, 请改语句8110和8138两语句, 其他语句可以不要改动。
2. 用键盘命令RUN启动本程序, 机器自动印出: "INPUT NOM. OF COMPLEX DATA N AND S", 询问原始序列的长度 ( $N=2^5$ )。
3. 由键盘回答 $N$ 和 $S$ 值之后, 立即印出原始数据序列的值和曲线。(若使用者不要印出这些内容, 删去8112句即可。) 然后印出所要的相关序列值和曲线。相关序列值为前两个值中的前一个值 (富里叶逆变换的实部)。

## 四、试题

$$\{x_r\} = \{y_r\} = \{\sin(2\pi f r)\}$$

$$f=2$$

$$r=0, 1, 2, \dots, N$$

$$\text{取 } N=16$$

## 五、程序清单及运行结果

### 程序清单

```

8000 GOTO 8092
8002 LET N1=N/2
8004 LET N2=N-1
8006 LET J=1
8008 FOR L=1 TO N2
8010 IF L=>J THEN GOTO 8024
8012 LET T1=A(2*J)
8014 LET T2=A(2*J+1)
8016 LET A(2*J)=A(2*L)
8018 LET A(2*J+1)=A(2*L+1)
8020 LET A(2*L)=T1
8022 LET A(2*L+1)=T2
8024 LET K=N1
8026 IF K=>J THEN GOTO 8034
8028 LET J=J-K
8030 LET K=K/2

```

```

8032 GOTO 8026
8034 LET J=J+K
8036 NEXT L
8038 LET P1=3.14159
8040 FOR I=1 TO S
8042 LET U1=0
8044 LET U2=0
8046 LET M=1
8048 FOR L=1 TO I
8050 LET M=M*2
8052 NEXT L
8054 LET K=M/2
8056 LET W1=COS(P1/K)
8058 LET W2=T*SIN(P1/K)
8060 FOR J=1 TO K
8062 FOR L=J TO N STEP M
8064 LET L1=L+K
8066 LET T1=A(2*L1)*U1-A(2*L1+1)*U2
8068 LET T2=A(2*L1)*U2+A(2*L1+1)*U1
8070 LET A(2*L1)=A(2*L)-T1
8072 LET A(2*L1+1)=A(2*L+1)-T2
8074 LET A(2*L)=A(2*L)+T1
8076 LET A(2*L+1)=A(2*L+1)+T2
8078 NEXT L
8080 LET Z=U1*W1-U2*W2
8082 LET U2=U1*W2+U2*W1
8084 LET U1=Z
8086 NEXT J
8088 NEXT I
8090 RETURN
8092 PRINT
8094 PRINT "INPUT NOM.OF COMPLEX DATA N AND S"
8096 INPUT N,S
8098 DIM A(2*N+2),B(2*N+2),C(2*N+2)
8100 PRINT
8102 LET X=3.14159/4
8104 PRINT "SERITES"
8106 PRINT
8108 FOR I=1 TO N/2
8110 LET A(2*I)=SIN((I-1)*X)

```

```

8112 PRINT 1-1;A(2*I);TAB(40+10*A(2*I));" + "
8114 LET A(2*I)=A(2*I)/N
8116 NEXT I
8118 PRINT
8120 LET T=-1
8122 GOSUB 8002
8124 FOR I=1 TO N
8126 LET B(2*I)=A(2*I)
8128 LET B(2*I+1)=A(2*I+1)
8130 LET A(2*I)=0
8132 LET A(2*I+1)=0
8134 NEXT I
8136 FOR I=1 TO N/2
8138 LET A(2*I)=SIN((1-I)*X)/N
8140 NEXT I
8142 LET T=-1
8144 GOSUB 8002
8146 FOR I=1 TO N
8148 LET C(2*I)=A(2*I)*B(2*I)+A(2*I+1)*B(2*I+1)
8150 LET C(2*I)=C(2*I)*N
8152 LET C(2*I+1)=A(2*I+1)*B(2*I)-A(2*I)*B(2*I+1)
8154 LET C(2*I+1)=C(2*I+1)*N
8156 NEXT I
8158 FOR I=1 TO N
8160 LET A(2*I)=C(2*I)/N
8162 LET A(2*I+1)=C(2*I+1)/N
8164 NEXT I
8166 GOSUB 8002
8168 FOR I=1 TO N
8170 PRINT 1-1; 2*A(2*I);2*A(2*I+1);TAB(40+10*A(2*I)*2);" . "
8172 NEXT I
8174 END

```

运行结果

INPUT NOM. OF COMPLEXE DATA N AND S

? 32? 5

SERITES

$$A(k)=B(k),$$

0	0	+			
1	.707106		+		
2	.999999			+	
3	.707109				+
4	4.52995 E -6			+	
5	-.707103		+		
6	-.999999				
7	-.707112		+		
8	-8.22544 E -6			+	
9	.707101				+
10	.999999				+
11	.707115			+	
12	1.28746 E -5				
13	-.707098		+		
14	-.999999	+			
15	-.707118		+		

实部( $R_{xy}(r)$ )

虚部

0	.500001	0			*
1	353553	4.54485 E -7			*
2	3.12505 E -2	-2.51457 E -7		*	
3	-.265166	-1.36346 E -6	*		
4	-.375	3.20375 E -7	*		
5	-.265166	-8.9407 E -8	*		
6	-3.12516 E -2	2.40282 E -7		*	
7	.176776	8.8264 E -7			*
8	.25	-6.85454 E -7			*
9	.176778	-2.84753 E -7			*
10	3.12515 E -2	-1.49016 E -8		*	
11	-8.83883 E -2	2.23517 E -7	*		
12	-.125	1.39326 E -6	*		
13	-.088389	8.9407 E -7	*		
14	-3.12505 E -2	-2.4214 E -8		*	
15	1.19209 E -7	-1.03563 E -6		*	

END AT 8174

此时印出其循环相关函数如下:

	实部	虚部
0	.500001	0
1	.353553	4.54485 E -7
2	3.12505 E -2	-2.51457 E -7
3	-.265166	-1.36346 E -6
4	-.375	3.20375 E -7
5	-.265166	-8.9407 E -8
6	-3.12516 E -2	2.40282 E -7
7	.176776	8.8264 E -7
8	.25	-6.85454 E -7
9	.176778	-2.84753 E -7
10	3.12515 E -2	-1.49016 E -8
11	-8.83883 E -2	2.23517 E -7
12	-.125	1.39326 E -6
13	-.088389	8.9407 E -7
14	-3.12505 E -2	-2.4214 E -8
15	1.19209 E -2	-1.03563 E -6
16	1.19209 E -7	0
17	8.9407 E -8	-3.72529 E -8
18	-3.12505 E -2	-3.4459 E -7
19	-8.83891 E -2	-1.22935 E -6
20	-.125	-3.20375 E -7
21	-8.83876 E -2	-3.27826 E -7
22	3.12516 E -2	3.55766 E -7
23	.176779	1.71016 E -6
24	.25	6.85454 E -7
25	.176776	7.01985 E -7
26	-3.12515 E -2	-5.81146 E -7
27	-.265167	-2.81632 E -6
28	-.375001	-1.39326 E -6
29	-.265164	-1.3113 E -6
30	3.12505 E -2	6.20261 E -7
31	.353555	3.62843 E -6

END AT 8174



### 参考文献

- [1] IEEE Transaction on Education Vol. 12, 1969  
NO. 1, 27—34

《The Fast Fourier Transform and its  
Application》

- [2] 清华大学工程力学系编译:

《随机振动和谱分析导论》

- [3] Brigham, E. Q. 著, 柳群译

《快速富里叶变换》

上海科学技术出版社 1979.3.

- [4] Robert K. Otnes, Loren Enochson

《Digital Time Series Analysis》

- [5] 程乾生编著

《信号数字处理的数学原理》

石油工业出版社 1979.

## 第十四章 线性代数计算

### §1 用高斯——约旦消去法求逆矩阵

#### 一. 方法概要:

本程序是对一个  $n$  阶的满秩实矩阵  $A$ , 进行有穷次的初等变换, 即用高斯——约旦消去法将其化成  $A^{-1}$ 。

在  $A$  矩阵的右侧, 加上一个  $n \times n$  的单位矩阵  $I$ , 得  $[A I]$ , 根据矩阵性质: 矩阵  $A$  经过有穷次初等变换化为矩阵  $B$ , 则  $A$  和  $B$  是等价的。  $[A I]$  经过初等变换后化为  $[I A^{-1}]$ , 两矩阵等价,  $\therefore A^{-1}$  即为所求的逆矩阵。

下面我们将  $[A I]$  作为一个矩阵的整体进行变换。

高斯——约旦消去法求逆矩阵的步骤:

1. 本方法主要是在一行上进行初等变换, 对  $n \times 2n$  的  $[A I]$  阵, 从第一列主元素开始, 寻找非零元素, 要求在主元素位置上不能为零元素, 如果在主元素位置上出现零元素, 则把这一列的任一非零元素换入, 同时将这两行的所有元素进行互换。

$$A(K, I) \Leftrightarrow A(L, I) \quad I=1, 2, \dots, 2 \times n$$

2. 非零主元素作为基, 如  $A(K, K)$ , 将主元变换成 1, 即:

$$A'_{KK} = \frac{A(K, K)}{A(K, K)} = 1$$

同时, 将本行的元素都除以  $A(K, K)$

$$A'(K, J) = A(K, J) / A(K, K) \quad \left( \begin{array}{l} J=1, 2, \dots, 2 \times n \\ J \neq K \end{array} \right) \quad (14-1-1)$$

3. 从第一列开始, 消除主元外的整个一列元素, 即:

$$A'(I, J) = A(I, J) - A(I, K) \cdot A(K, J) / A(K, K)$$

$$= A(I, J) - A(I, K) \cdot A'(K, J)$$

$$(I=1, 2, \dots, N, J=1, 2, \dots, 2 \times N)$$

$$(I, J \neq K)$$

$$(14-1-2)$$

使这一列除主元以外都为 0, 同时处理一行。

4. 是否做完, 若已化成  $[I A^{-1}]$  阵, 则计算结束, 否则返回步骤 1, 重复进行。  
对于  $[A I]$  具体的计算如下:

$$[A I] = \left( \begin{array}{cccccc} a_{11} & a_{12} & \dots & a_{1n} & 1 & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & a_{2n} & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & 0 & 0 & \dots & 1 \end{array} \right) \rightarrow \left( \begin{array}{cccccc} 1 & a_{12}^{(1)} & \dots & a_{1n}^{(1)} & 1/a_{11} & 0 & \dots & 0 \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} & -a_{21}/a_{11} & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} & -a_{n1}/a_{11} & \dots & \dots & 1 \end{array} \right)$$

$$\rightarrow \left( \begin{array}{cccccc} 1 & 0 & \dots & 0 & a_{1n+1}^{(n)} & a_{1n+2}^{(n)} & \dots & a_{12 \times n}^{(n)} \\ 0 & 1 & \dots & 0 & a_{2n+1}^{(n)} & a_{2n+2}^{(n)} & \dots & a_{22 \times n}^{(n)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & a_{nn+1}^{(n)} & a_{nn+2}^{(n)} & \dots & a_{n2 \times n}^{(n)} \end{array} \right)$$

以上右半部分的内容就是 $A^{-1}$ 。

## 二、操作说明

### (一) 本程序使用的工作单元

N 方阵A的阶数

A 满秩实矩阵, 开始存放求逆的满秩方阵, 后来加入单位矩阵I, 为 $N \times 2N$ 矩阵, 最后存放[I]元素和 $A^{-1}$ 元素,

I, J, S, T为循环变量

X 简单变量

### (二) 操作步骤:

1. 由BASIC解释程序将本程序送入机内
2. 由键盘命令RUN启动本程序执行。计算机自动在电传机上打印出:  
"INPUT N", 询问给定矩阵的维数N, 用户根据欲求逆的矩阵的最大维数进行回答。
3. 由键盘输入求逆矩阵的维数实际值N, 电传机又打出:"INPUT MATRIX"
4. 由键盘输入A矩阵的各个元素, 其格式是一行一行的输入, 每输入一个元素必须按“回车”键, 输完一行, 电传机自动回车换行, 这样一行一行输入, 直至输入完毕, 输入完毕机器自动进行计算, 最后打印出计算结果 $A^{-1}$ 。
5. 输出格式, 一行只打印一个数据, 连续打印出一列数据, 此数据就是 $A^{-1}$ 的第一列, 然后空一行, 再接着打印输出, 直至结束, 如果, A矩阵奇异, 则计算失败, 打印出  
"MATRIX SINGULAR"
6. 计算结束, 机器打出问话:"MORE INPUT (1=YES, 0=NO)"  
用户必须作出回答, 若回答“1”则程序自动重新启动, 重新计算, 若回答“0”则程序结束。

### 7. 例题:

$$\text{已知: } A = \begin{pmatrix} 3 & -6 & 1 & 0 \\ 3 & 0 & 0 & 1 \\ 2 & -1 & 1 & 0 \\ 3 & -1 & 1 & 0 \end{pmatrix} \quad \text{则 } A^{-1} = \begin{pmatrix} 0 & 1 & -1 & 1 \\ -0.2 & 0 & 0 & 0.2 \\ -0.2 & 0 & 3 & -0.8 \\ 0 & 0 & 3 & 0 \end{pmatrix}$$

## 三、程序清单

```
6000 REM INVER MATRIX PROGRAM
6002 PRINT "INPUT N"
6004 INPUT N
6006 DIM A(N,2*N)
6008 PRINT
6010 PRINT "INPUT MATRIX A"
6012 FOR S=1 TO N
6014   FOR J=1 TO N
6016     INPUT A(S,J)
6018   IF J=S THEN GOTO 6024
```

```

6020     LET A(S, J+N)=0
6022     GOTO 6026
6024     LET A(S, J+N)=1
6026     NEXT J
6028     PRINT
6030     NEXT S
6032     PRINT
6034     FOR S=1 TO N
6036     FOR T=S TO N
6038     IF A(T,S)><0 THEN GOTO 6046
6040     NEXT T
6042     PRINT "MATRIX SINGULAR"
6044     GOTO 6114
6046     GOSUB 6066 ✓
6048     LET C=1/A(S,S)
6050     GOSUB 6078
6052     FOR T=1 TO N
6054     IF T=S IHEN GOTO 6060
6056     LET C=-A(T,S)
6058     GOSUB 6086
6060     NEXT T
6062     NEXT S
6064     GOTO 6094
6066     FOR J=1 TO 2*N
6068     LET B=A(S,J)
6070     LET A(S,J)=A(T,J)
6072     LET A(T,J)=B
6074     NEXT J
6076     RETURN
6078     FOR J=1 TO 2*N
6080     LET A(S,J)=C*A(S,J)
6082     NEXT J
6084     RETURN
6086     FOR J=1 TO 2*N
6088     LET A(T,J)=A(T,J)+C*A(S,J)
6090     NEXT J
6092     RETURN
6094     PRINT
6096     FOR J=N+1 TO 2*N
6098     FOR I=1 TO N

```

```

6100     PRINT A(I,J)
6102     NEXT I
6104     PRINT
6106     NEXT J
6108     PRINT "MORE INPUT (1=YES, 0=NO)"
6110     INPUT X
6112     IF X=1 THEN GOTO 6002
6114     END

```

四 程序运行

RUN

INPUT N

? 4

INPUT MATRIX A

? 3 ? -6 ? 1 ? 0

? 3 ? 0 ? 0 ? 1

? 2 ? -1 ? 1 ? 0

? 3 ? -1 ? 1 ? 0

-1.19209E-8

-.2

-.2

-1.78814E-7

0

0

0

1

- 1

0

3

3

1

.2

-1.8

- 3

MORE INPUT (1=YES, 0=NO)

? 0

END AT 6114

## §2 高斯——约旦迭代法求逆矩阵

### 一、算法概要

本程序利用高斯——约旦迭代法直接求出A的逆矩阵 $A^{-1}$ 。

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \rightarrow \begin{pmatrix} 1/a_{11} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ a_{21}^{(2)} & 1/a_{22} & \cdots & a_{2n}^{(2)} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1}^{(2)} & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix} \rightarrow \cdots \rightarrow \begin{pmatrix} a_{11}^{(n)} & a_{12}^{(n)} & \cdots & a_{1n}^{(n)} \\ a_{21}^{(n)} & a_{22}^{(n)} & \cdots & a_{2n}^{(n)} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1}^{(n)} & a_{n2}^{(n)} & \cdots & 1/a_{nn} \end{pmatrix}$$

计算步骤:

1. 从第一列开始, 如果主元素为非零元素则开始迭代计算, 如果为零元素, 则在同一列上找到第一个非零元素, 并进行这两行的互换, 并记下非零元素所在的行号。
2. 主元换成 $A^{(k)}(S, S) = 1/A^{(k-1)}(S, S)$ , 同时这一行全部乘上 $1/A^{(k-1)}(S, S)$   
即:  $A^{(k)}(S, J) = A^{(k-1)}(S, J)/A^{(k-1)}(S, S) = A^{(k-1)}(S, J) \times A^{(k)}(S, S)$   
( $J=1, 2, \dots, N$ ) ( $J \neq S$ ) (14-2-1)
3. 将一行进行迭代计算  
 $A^{(k+1)}(I, J) = A^{(k)}(I, J) - A^{(k)}(I, S) \times A^{(k)}(S, J)$   
( $I, J=1, 2, \dots, N; I, J \neq S$ ) (14-2-2)
4. 如果迭代结束, 即迭代已到N次, 则迭代过程结束。此时, 由于前面出现的行互换, 再进行一次列互换, 如果主元没有进行交换, 则不进行交换。如果迭代过程没结束, 则返回1。

### 二、操作说明

(一) 本程序使用的工作单元有:

N 输入满秩矩阵的阶数

A 开始为输入矩阵, 程序执行中间为进行迭代运算的结果, 最后存放 $A^{-1}$ 的各元素。

V 记录换入主元的非零元素的行号

B 中间变量, S, T, J 为循环变量

(二) 操作步骤:

1. 由BASIC解释程序将本程序送入计算机内。
2. 用键盘命令 RUN 使本程序投入运行。
3. 电传机打印出: "ENTER N" 用户用键盘回答A矩阵的实际维数。
4. 机器又打印出: "ENTER MATRIX A"
5. 由键盘输入A矩阵的各个元素, 其格式是一行一行地输入, 每输入一个元素必须按

“回车”键。输入完一行。电传机自动回车换行。输入第二行的元素。直至全部输入完毕。

6. 机器开始计算。打印出计算结果 $A^{-1}$ 。如果A矩阵为奇异。则打印出。

“MATRIX SINGULAR”

在计算完后输出 $A^{-1}$ 的结果时。按列分块输出。每一行只打印一个数据。每一块为一列。块与块之间空一行。直至全部输出。

7. 计算结束后。机器打出问话。

“MORE INPUT (1=YES, 0=NO)”

用户须作出回答。若回答“1↵”,则该程序重新启动计算,若回答“0↵”,则该程序结束。

### 三、例题

$$A = \begin{pmatrix} 1 & 10 & 1 \\ 2 & 0 & 1 \\ 3 & 3 & 2 \end{pmatrix}$$

计算结果:

$$A^{-1} = \begin{pmatrix} 0.428572 & 2.42857 & -1.42857 \\ 0.142857 & 0.142857 & -0.142857 \\ -0.857144 & -3.85714 & 2.85714 \end{pmatrix}$$

### 四、程序清单及运行结果

#### 程序清单

```
6118 REM COMPUTING THE INVERSE MATRIX*
6120 PRINT "ENTER N"
6122 INPUT N
6124 PRINT
6126 DIM A(N,N),V(N)
6128 PRINT "ENTER MATRIX A"
6130 FOR S=1 TO N
6132   FOR J=1 TO N
6134     INPUT A(S,J)
6136   NEXT J
6138   PRINT
6140 NEXT S
6142 FOR S=1 TO N
6144   FOR T=S TO N
6146     IF A(T,S)><0 THEN GOTO 6154
6148   NEXT T
6150   PRINT "MATRIX SINGULAR"
6152   GOTO 6246
6154   GOSUB 6192
6156   LET A(S,S)=1/A(S,S)
6158   GOSUB 6206
6160   FOR T=1 TO N
6162     IF T=S THEN GOTO 6170
```

```

6164     LET B=-A(T,S)
6166     LET A(T,S)=0
6168     GOSUB 6216
6170     NEXT T
6172     NEXT S
6174     FOR S=N TO 1 STEP -1
6176     IF V(S)=S THEN GOTO 6188
6178     FOR J=1 TO N
6180         LET B=A(J,S)
6182         LET A(J,S)=A(J,V(S))
6184         LET A(J,V(S))=B
6186     NEXT J
6188     NEXT S
6190     GOTO 6224
6192     FOR J=1 TO N
6194         LET B=A(S,J)
6196         LET A(S,J)=A(T,J)
6198         LET A(T,J)=B
6200     NEXT J
6202     LET V(S)=T
6204     RETURN
6206     FOR J=1 TO N
6208         IF J=S THEN GOTO 6212
6210         LET A(S,J)=A(S,S)*A(S,J)
6212     NEXT J
6214     RETURN
6216     FOR J=1 TO N
6218         LET A(T,J)=A(T,J)+B*A(S,J)
6220     NEXT J
6222     RETURN
6224     PRINT "OUTPUT INVERSE MATRIX A"
6226     FOR J=1 TO N
6228         PRINT
6230         FOR I=1 TO N
6232             PRINT A(I,J)
6234         NEXT I
6236         PRINT
6238     NEXT J
6240     PRINT "MORE INPUT(1=YES, 0=NO)"
6242     INPUT X

```



```
6244 IF X=1 THEN GOTO 6120
```

```
6246 END
```

程序运行结果:

```
RUN
```

```
ENTER N
```

```
? 3
```

```
ENTER MATRIX A
```

```
? 1 ? 10 ? 1
```

```
? 2 ? 0 ? 1
```

```
? 3 ? 3 ? 2
```

```
OUTPUT INVERSE MATRIX A
```

```
.42857
```

```
.142857
```

```
— .85714
```

```
2.42857
```

```
.142857
```

```
— 3.85714
```

```
—1.42857
```

```
— .142857
```

```
2.85714
```

```
MORE INPUT(1=YES, 0=NO)
```

```
? 0
```

```
END AT 6246
```

### §3 矩 阵 相 乘

#### 一、计算方法概要

本程序用来将  $N \times M$  的 A 矩阵与右边的  $M \times P$  的 B 矩阵相乘, C 为矩阵的乘积, 根据矩阵相乘规则, C 矩阵的每一个元素分别按下式计算得到。

$$C(I,J) = \sum_{k=1}^M A(I,K) * B(K,J) \quad (14-3-1)$$

$$\begin{pmatrix} I=1, 2, \dots, N \\ J=1, 2, \dots, P \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \dots & & & \\ b_{m1} & b_{m2} & \dots & b_{mp} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1p} \\ c_{21} & c_{22} & \dots & c_{2p} \\ \dots & & & \\ c_{n1} & c_{n2} & \dots & c_{np} \end{pmatrix}$$

#### 二、操作说明

##### (一) 本程序所用的工作单元

A(N,M) 相乘的矩阵

B(M,P) 相乘的矩阵  
C(N,P) 乘积矩阵  
S, I, J, K, X 为工作单元

## (二) 操作步骤:

1. 由BASIC解释程序将本程序送入机内。
2. 由键盘发命令 RUN 启动本程序执行。电传机打印出:  
"INPUT DIMENSIONS N, M, P"  
用户将按参加运算的矩阵中的维数分别给 N, M, P 赋值, N 为被乘矩阵的行数, M 为被乘矩阵的列数及乘矩阵的行数, P 为乘矩阵的列数。每输入一个数据要按回车键。
3. 回答完以上值后, 打印出:  
"INPUT MATRIX A"  
用户输入 A 矩阵。按行输入, 一行输入完毕, 自动换行, 每输入一个元素都要以回车键结束直至全部数据输入完毕。
4. 全部输入后, 电传打印出:  
"INPUT MATRIX B"  
用户输入 B 矩阵, 输入方法同 4。
5. A, B 都输入后, 计算机开始计算, 最后打印出:  
"OUTPUT MULTIPLY RESULT"  
将 C 矩阵按列分组打印全部结果。第一组数为第一列元素, 第二组为第二列元素。
6. 输出结果后, 电传打印出:  
"MORE INPUT (1=YES, 0=NO)"  
用户可根据需要进行回答: 回答 1 时, 本程序将重新投入运行; 回答 0 时, 本程序运行结束。

## 三、例题

$$A \times B = \begin{pmatrix} 1 & 2 & -1 & 4 \\ 5 & 2 & 1 & 0 \\ 3 & -1 & -2 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & -1 & -2 & 0 \\ 2 & -1 & 0 & 0 & 3 \\ -8 & 7 & 1 & 1 & -1 \\ -3 & 4 & 1 & 0 & 7 \end{pmatrix}$$

计算结果:

$$A \times B = \begin{pmatrix} 1 & 8 & 2 & 1 & 35 \\ 1 & 10 & -4 & 11 & 5 \\ 20 & -14 & -6 & 4 & -8 \end{pmatrix}$$

## 四、程序清单及运行结果

### 程序清单

```
6200 REM MATRIX MULTIPLING
6202 PRINT "INPUT MATRIX DIMENSIONS N,M,P"
6204 PRINT "INPUT N,M,P"
6206 INPUT N,M,P
6208 PRINT
6210 DIM A(N,M),B(M,P),C(N,P)
6212 PRINT "INPUT MATRIX A"
6214 FOR I=1 TO N
```

```

6216   FOR J=1 TO M
6218       INPUT A(I,J)
6220   NEXT J
6222   PRINT
6224 NEXT I
6226 PRINT "INPUT MATRIX B"
6228 PRINT
6230 FOR I=1 TO M
6232     FOR J=1 TO P
6234         INPUT B(I,J)
6236     NEXT J
6238     PRINT
6240 NEXT I
6242 FOR I=1 TO N
6244     FOR J=1 TO P
6246         LET C(I,J)=0
6248         FOR K=1 TO M
6250             LET C(I,J)=C(I,J)+A(I,K)*B(K,J)
6252         NEXT K
6254     NEXT J
6256 NEXT I
6258 PRINT "OUTPUT RESULT MATRIX"
6260 FOR J=1 TO P
6262     FOR I=1 TO N
6264         PRINT C(I,J)
6266     NEXT J
6268 PRINT
6270 NEXT I
6272 PRINT "MORE INPUT(1=YES,0=NO)"
6274 INPUT X
6276 IF X=1 THEN GOTO 6212
6278 END

```

程序运行

RUN

INPUT MATRIX DIMENSIONS N, M, P

INPUT N, M, P

? 3 ? 4 ? 5

INPUT MATRIX A

? 1 ? 2 ? -1 ? 4

? 5 ? 2 ? 1 ? 0

```

? 3 ? -1 ? -2 ? -1
INPUT MATRIX B
? 1 ? 1 ? -1 ? 2 ? 0
? 3 ? -1 ? 0 ? 0 ? 3
? -8 ? 7 ? 1 ? 1 ? -1
? -3 ? 4 ? 1 ? 0 ? 7
OUTPUT RESULT MATRIX
1
1
20
8
10
-14
2
-4
-6
1
11
4
35
5
-8
MORE INPUT (1=YES,0=NO)
? 0
END AT 6278

```

#### §4 求行列式的值

##### 一 计算方法概述

定义：对于  $n$  阶行列式

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = \sum_{j_1, j_2, \dots, j_n} (-1)^{r(j_1, j_2, \dots, j_n)} a_{1j_1} a_{2j_2} \cdots a_{nj_n} \quad (14-4-1)$$

这里  $\sum_{j_1, j_2, \dots, j_n}$  表示对所有  $n$  级排列求和

根据以上定义可得出：

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ 0 & 0 & \cdots & 0 & a_{nn} \end{vmatrix} = a_{11} a_{22} a_{33} \cdots a_{nn} \quad (14-4-2)$$

也即上三角形行列式的值为主对角线上元素的乘积。

$n$  级行列式具有以下性质:

1. 在行列式中, 将一行的倍数加到另一行, 行列式值不变。
2. 在行列式中, 将两行或两列的位置互换, 则行列式的值反号。

根据以上两条性质, 我们可以把一个任意实数  $n$  级行列式变换成上三角形行列式, 来求出行列式的值。

一般按以下步骤进行:

1. 从第一行开始, 检查主元素是否为 0, 若为 0, 则在同一行的不同列上找非零元素, 找到后, 互换两列, 使主元素不为零, 同时, 行列式的符号要反号, 即  $S = -S$
2. 根据性质 1, 将主元素所在列中的以下元素变换成 0, 同时, 其他列的元素作相应的变换, 变换关系为:

$$H = A(J, I) / A(I, I) \quad \begin{pmatrix} I=1, 2, \dots, N-1 \\ J=I+1, I+2, \dots, N \end{pmatrix} \quad (14-4-3)$$

$$A(J, K) = A(J, K) - A(I, K) * H \quad \begin{pmatrix} K=I, I+1, \dots, N-1 \\ I=1, 2, \dots, N \\ J=I+1, I+2, \dots, N \end{pmatrix} \quad (14-4-4)$$

3. 将主对角线元素相乘, 即:

$$D = D * A(I, I) \quad (I = 1, 2, \dots, N) \quad (14-4-5)$$

$$D = S * D \quad (14-4-6)$$

## 二、操作说明

(一) 本程序使用的工作单元为:

$A(N, N)$  输入行列式, 经过变换后, 变成上三角形矩阵。

$D$  为行列式的值

$S$  行列式的符号

$H$  中间计算变量

$M$  用户所需定维数

$N$  行列式的阶数

$E$  用户所定的最小正数

$I, J, K$  为循环变量

(二) 操作步骤

1. 由BASIC解释程序将本程序送入机内。
2. 用键盘命令RUN↵, 使本程序投入运行, 电传机打印出:

"INPUT DIMENSIONS M"

要求用户用正整数回答, 若用户要计算若干个行列式的值, 则应挑选阶数最高的行列式来定维。若只计算一个行列式, 则就用本行列式的阶数来定维。

3. 回答完维数后, 电传又打印出:

"INPUT ORDERS OF A"

向用户询问行列式的阶数, 行列式是  $n$  阶就回答  $n$ 。

4. 回答完阶数后, 电传又打印出:

"INPUT REAL DETERMINANT A"

要求用户输入行列式的各元素，按行输入，输入一行后，自动回车，接着输入第二行的各元素，直至输入结束。

5. 输入完行列式各元素后，电传又打印出：

"INPUT SMALLEST POSITIVE NUMBER"

这时，用户可根据精确度要求，输入一个最小的正数，程序中凡是发现一个元素的绝对值小于此正数，就作为 0 处理。

6. 计算机开始计算，计算完毕后，打印出：

"OUTPUT THE VALUE OF DETERMINANT"

D = \* \* \* \*

7. 计算结束后，电传又问

"MORE INPUT (1=YES, 0=NO)"

用户用 "1" 或 "0" 来回答，若回答 1，则重新启动本程序，从步骤 3 开始运行。若回答 0 则结束运行，退出本程序。

### 三、例题

$$A = \begin{vmatrix} 3 & 2 & 4 & 1 \\ 2 & 0 & 2 & 5 \\ 2 & 1 & 2 & 1 \\ 2 & 4 & 3 & 1 \end{vmatrix} = 21$$

### 四、程序清单及运行结果

#### 程序清单

```
6300 REM THE PROGRAM OF COMPUTING THE VALUE OF
6302 REM REAL DETERMINANT
6304 PRINT "INPUT DIMENSIONS M"
6306 INPUT M
6308 DIM A(M,M)
6310 PRINT
6312 PRINT "INPUT THE ORDERS OF A"
6314 INPUT N
6316 PRINT
6318 PRINT "INPUT REAL DETERMINANT A"
6320 FOR I=1 TO N
6322 FOR J=1 TO N
6324 INPUT A(I,J)
6326 NEXT J
6328 PRINT
6330 NEXT I
6332 PRINT
6334 PRINT "INPUT SMALLEST POSITIVE NUMBER"
6336 INPUT E
6338 PRINT
```

```

6340 LET D=1
6342 LET S=1
6344 FOR I=1 TO N-1
6346   FOR J=1 TO N
6348     IF ABS(A(I,J))>E THEN GOTO 6356
6350   NEXT J
6352   LET D=0
6354   GOTO 6388
6356   IF I=J THEN GOTO 6370
6358   FOR K=1 TO N
6360     LET A(0,0)=A(K,I)
6362     LET A(K,1)=A(K,J)
6364     LET A(K,J)=A(0,0)
6366   NEXT K
6368   LET S=-S
6370   FOR J=I+1 TO N
6372     LET H=A(J,I)/A(I,I)
6374     FOR K=1 TO N
6376       LET A(J,K)=A(J,K)-A(I,K)*H
6378     NEXT K
6380   NEXT J
6382   LET D=D*A(I,I)
6384 NEXT I
6386 LET D=D*A(N,N)
6388 PRINT "OUTPUT THE VALUE OF DETERMINANT"
6390 PRINT "D=";D
6392 PRINT "MORE INPUT (1=YES,0=NO)"
6394 INPUT X
6396 IF X=1 THEN GOTO 6312
6398 END

```

程序运行结果:

RUN

INPUT DIMENSIONS M

? 4

INPUT THE ORDERS OF A

? 4

INPUT REAL DETERMINANT A

? 3 ? 2 ? 4 ? 1

? 2 ? 0 ? 2 ? 5

? 2 ? 1 ? 2 ? 1

```

? 2 ? 4 ? 3 ? 1
INPUT SMALLEST POSITIVE NUMBER
? .00001
OUTPUT THE VALUE OF DETERMINANT
D=21
MORE INPUT (1=YES,0=NO)
? 0
END AT 6398

```

## §5 求实对称矩阵特征值和特征向量

### 一、算法概要

任一个  $n$  阶实对称矩阵  $A$  可经一系列的初等旋转变换（正交变换）化为对角阵  $D$ ， $D$  的对角线元素即为  $A$  的特征值，其变换关系式如下：

$$A_{k+1} = U_k^T A_k U_k \quad (A_k = A \text{ 为已知输入矩阵})$$

其中  $U_k = U_k(p, q, \phi)$  为正交旋转阵，其形式为：

$$U_k = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \cos\phi & \sin\phi & \\ & & & \vdots & \vdots & \\ & & & -\sin\phi & \cos\phi & \\ & & 0 & & & 1 & \ddots & \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \leftarrow \text{第 } p \text{ 行} \\ \leftarrow \text{第 } q \text{ 行} \end{matrix}$$

$\uparrow \quad \uparrow$   
 第  $p$  列 第  $q$  列

$$U_k^T = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \cos\phi & -\sin\phi & \\ & & & \vdots & \vdots & \\ & & & \sin\phi & \cos\phi & \\ & & 0 & & & 1 & \ddots & \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \leftarrow \text{第 } p \text{ 行} \\ \leftarrow \text{第 } q \text{ 行} \end{matrix}$$

$\uparrow \quad \uparrow$   
 第  $p$  列 第  $q$  列

也即  $U_k$  的元素组成是： $U_{pp} = \cos\phi = U_{qq}$

$U_{pq} = -U_{qp} = \sin\phi$ ，除此之外的所有元素与  $n$  阶单位矩阵相同。

其中， $(p, q = 1, 2, \dots, n)$   
 $p < q$

$A_{k+1}$  是  $A_k$  进行一次旋转变换后所得矩阵，因此  $A_{k+1}$  中的第  $p, q$  行和第  $p, q$  列上的元素与  $A_k$  的相应元素是不同的。

$$\therefore A_{k+1} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & \cos\phi & -\sin\phi & \\ & & \sin\phi & \cos\phi & \\ & & & & 1 & \ddots & \\ & & 0 & & & & 1 \end{pmatrix} \times \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & & \\ \dots & & a_{pp} & a_{pq} & a_{pn} \\ \dots & & a_{qp} & a_{qq} & a_{qn} \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \times \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & \cos\phi & \sin\phi & \\ & & -\sin\phi & \cos\phi & \\ & & & & 1 & \ddots & \\ & & 0 & & & & 1 \end{pmatrix}$$



$A_{k+1}$  阵中的  $a_{pp}^{(k+1)}$ ,  $a_{pq}^{(k+1)}$ ,  $a_{qp}^{(k+1)}$ ,  $a_{qq}^{(k+1)}$  分别为:

$$\begin{cases} a_{pp}^{(k+1)} = \cos^2 \phi \times a_{pp}^{(k)} - 2\cos \phi \sin \phi \times a_{pq}^{(k)} + \sin^2 \phi \times a_{qq}^{(k)} & (14-5-1) \\ a_{qq}^{(k+1)} = \sin^2 \phi \times a_{pp}^{(k)} + 2\cos \phi \sin \phi \times a_{pq}^{(k)} + \cos^2 \phi \times a_{qq}^{(k)} & (14-5-2) \\ a_{pq}^{(k+1)} = a_{qp}^{(k+1)} = (\cos^2 \phi - \sin^2 \phi) a_{pq}^{(k)} + \cos \phi \sin \phi (a_{pp}^{(k)} - a_{qq}^{(k)}) & (14-5-3) \end{cases}$$

( $\because A$  是对称阵)

当  $j \neq p, q$  时:

$$\begin{cases} a_{pj}^{(k+1)} = \cos \phi \times a_{pj}^{(k)} - \sin \phi \times a_{qj}^{(k)} & (j=1, 2, \dots, N) \\ a_{qi}^{(k+1)} = \sin \phi \times a_{pi}^{(k)} + \cos \phi \times a_{qi}^{(k)} & (j \neq p, q) \end{cases} \quad \begin{matrix} (14-5-4) \\ (14-5-5) \end{matrix}$$

为了得到  $A_{k+1}$  阵, 根据以上公式主要是确定  $\phi$  值, 首先我们来求出  $\phi$  值. 根据正交旋转变换的目的, 除了主元以外, 其他所有元素都应变换成零元素, 所以我们可以令  $a_{pq}^{(k+1)} = 0$ , 即:

$$(\cos^2 \phi - \sin^2 \phi) a_{pq}^{(k)} + \cos \phi \sin \phi (a_{pp}^{(k)} - a_{qq}^{(k)}) = 0$$

移项得:

$$(\cos^2 \phi - \sin^2 \phi) a_{pq}^{(k)} = -\cos \phi \sin \phi (a_{pp}^{(k)} - a_{qq}^{(k)})$$

$$2(\cos^2 \phi - \sin^2 \phi) a_{pq}^{(k)} = -2\cos \phi \sin \phi (a_{pp}^{(k)} - a_{qq}^{(k)})$$

$$\text{即满足 } \text{tg} 2\phi = 2a_{pq}^{(k)} / (a_{pp}^{(k)} - a_{qq}^{(k)})$$

为避免分母为零:

$$\text{ctg} 2\phi = \frac{(a_{pp}^{(k)} - a_{qq}^{(k)})}{2a_{pq}^{(k)}} \quad (14-5-6)$$

根据三角公式有:

$$\sin 2\phi = 1 / \sqrt{1 + \text{ctg}^2 2\phi}$$

$$\sin \phi = \sin 2\phi / \sqrt{2(1 + \sqrt{1 - \sin^2 2\phi})} \quad (14-5-7)$$

$$\cos \phi = \sqrt{1 - \sin^2 \phi} \quad (14-5-8)$$

有了以上的基本计算公式, 就可以确定计算步骤:

1. 求出旋转变换参数.

$$M_1 = \frac{1}{2} [A(p, p) - A(q, q)]$$

$$\sin 2\phi = W = \begin{cases} -A(p, q) / \sqrt{A(p, q)^2 + [(A(p, p) - A(q, q))/2]^2} & M_1 > 0 \\ +A(p, q) / \sqrt{A(p, q)^2 + [(A(p, p) - A(q, q))/2]^2} & M_1 < 0 \end{cases}$$

$$\sin \phi = S = W / \sqrt{2(1 + \sqrt{1 - (W)^2})} \quad (14-5-9)$$

$$\cos \phi = C = \sqrt{1 - S^2} \quad (14-5-10)$$

2. 计算, 即进行旋转变换, 求得相应的新元素

$$A^{(k+1)}(p, p) = CA^{(k)}(p, p) - 2CSA^{(k)}(p, q) + S^2A^{(k)}(q, q) \quad (q > p) \quad (14-5-11)$$

$$A^{(k+1)}(q, q) = S^2A^{(k)}(p, p) + 2CSA^{(k)}(p, q) + C^2A^{(k)}(q, q) \quad (14-5-12)$$

$$A^{(k+1)}(p, q) = (C^2 - S^2)A^{(k)}(p, q) + CS[A^{(k)}(p, p) - A^{(k)}(q, q)] \quad (14-5-13)$$

3. 消去下三角形中各元素

$$A^{(k+1)}(i, p) = CA^{(k)}(i, p) - SA^{(k)}(i, q) \quad (14-5-14)$$

$$A^{(k+1)}(i, q) = SA^{(k)}(i, p) + CA^{(k)}(i, q) \quad (i \neq p, q) \quad (14-5-15)$$

$$i = 1, 2, \dots, N$$

4. 消去上三角元素或下三角元素以后, 对角线元素即为特征值。

5. 计算特征向量:

取单位矩阵 $S(i, i)$ 计算新的 $S(i, j)$ 的各元素, 关系式如下:

$$S(i, p) = S(i, p)C - S(i, q)S \quad (14-5-16)$$

$$S(i, q) = S(i, p)S + S(i, q)C \quad (14-5-16)$$

6. 计算出的新矩阵 $S(i, j)$ , 其中各列元素对应各个特征值的特征向量。

## 二、程序说明

本程序使用的工作单元有:

$A(I, J)$  输入实对称矩阵, 最后 $A(I, I)$  为特征值。

$S(I, J)$  开始为单位阵, 最后得到的矩阵为特征向量矩阵, 每一列对应一个特征值的特征向量。

$C_1, T_1$  分别表示 $\sin\phi$ 和 $\cos\phi$

$V_1, V_2, V_3$  为计算中间变量。

$P, Q, I, J$  为循环变量。

## 三、上机操作步骤

1. 用BASIC解释程序将本程序送入机内。

2. 由用户发键盘命令RUN启动本程序执行, 则电传机打印出:

"INPUT N"

用户回答求逆矩阵的实际维数。

3. 电传又打印出:

"INPUT MATRIX A"

用户输入要求特征值和特征向量的矩阵A, 按行输入, 每输入一个数据都以“回车”键结束, 直到全部输入为止。

4. 计算机开始计算, 计算完后打印出:

"EIGENVALUE EIGENVECTOR"

接着输出一个特征值, 一组特征向量的元素, 直到全部输出为止

5. 电传又打印出问话

"MORE INPUT (1=YES, 0=NO)"

用户可根据需要回答, 若不要进行计算了就回答0, 若要计算, 就回答1, 计算机在回答1的情况下, 回到第三步操作, 回答零时, 运行结束。

#### 四、例题

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

计算结果

特征值	特征向量
2.44949	0.673888
	-0.302905
	0.673887
-2.44949	0.214186
	0.953021
	0.214186
2	-0.707106
	-2.4047E-7
	0.707107

#### 五、程序清单及运行结果

程序清单

```

6280 REM THE JACOBI METHOD FOR COMPUTING EIGENVALUE AND
6282 REM EIGENVECTOR
6284 PRINT "INPUT N"
6286 INPUT N
6288 PRINT
6290 DIM A(N,N),S(N,N)
6292 PRINT "INPUT MATRIX A"
6294 FOR I=1 TO N
6296   FOR J=1 TO N
6298     INPUT A(I,J)
6300     IF I=J THEN GOTO 6306
6302     LET S(I,J)=0
6304     GOTO 6308
6306     LET S(I,J)=1
6308   NEXT J
6310   PRINT
6312 NEXT I
6314 LET R=.00001
6316 FOR J=2 TO N
6318   FOR I=1 TO J-1
6320     LET I1=I+2*A(I,J)*A(I,J)
6322   NEXT I
6324 NEX J

```

```

6326 LET N1=SQR(I1)
6328 LET N2=(R/N) * N1
6330 LET T=N1
6332 LET T=T/N
6334 FOR Q=2 TO N
6336   FOR P=1 TO Q-1
6338     IF ABS(A(P,Q))=<T THEN GOTO 6400
6340     LET I2=1
6342     LET V1=A(P,P)
6344     LET V2=A(P,Q)
6346     LET V3=A(Q,Q)
6348     LET M1=(V1-V3) * .5
6350     IF M1><0 THEN GOTO 6356
6352     LET W=-1
6354     GOTO 6358
6356     LET W=-SGN(M1) * V2/SQR(V2 * V2 + M1 * M1)
6358     LET T1=W/SQR(2 * (1+SQR(1-W ↑ 2)))
6360     LET T2=T1 * T1
6362     LET C1=SQR(1-T2)
6364     LET C2=C1 * C1
6366     LET T3=T1 * C1
6368     FOR I=1 TO N
6370       LET I1=A(I,P) * C1 - A(I,Q) * T1
6372       LET A(I,Q)=A(I,P) * T1 + A(I,Q) * C1
6374       LET A(I,P)=I1
6376       LET I1=S(I,P) * C1 - S(I,Q) * T1
6378       LET S(I,Q)=S(I,P) * T1 + S(I,Q) * C1
6380       LET S(I,P)=I1
6382     NEXT I
6384     FOR I=1 TO N
6386       LET A(P,I)=A(I,P)
6388       LET A(Q,I)=A(I,Q)
6390     NEXT I
6392     LET A(P,P)=V1 * C2 + V3 * T2 - 2 * V2 * T3
6394     LET A(Q,Q)=V1 * T2 + V3 * C2 + 2 * V2 * T3
6396     LET A(P,Q)=(C2 - T2) * V2 + T3 * (V1 - V3)
6398     LET A(Q,P)=A(P,Q)
6400   NEXT P
6402 NEXT Q
6404 IF I2><1 THEN GOTO 6410

```

```

6406 LET I2= 0
6408 GOTO 6334
6410 IF T>N2 THEN GOTO 6332
6412 PRINT 'EIGENVALUE', 'EIGENVECTOR'
6414 PRINT
6416 FOR I=1 TO N
6418   PRINT A(I,I),S(1,I)
6420   FOR J=2 TO N
6422     PRINT ,S(J,I)
6424   NEXT J
6426   PRINT
6428   PRINT
6430 NEXT I
6432 PRINT 'MORE INPUT(1=YES, 0=NO)'
6434 INPUT X
6436 IF X=1 THEN GOTO 6284
6438 END

```

程序运行结果:

RUN

INPUT N

? 3

INPUT MATRIX A

? 2? -1? 0

? -1? -2? -1

? 0? -1? 2

EAGENVALUE      EIGENVECTOR

2.44949            .673888

                 - .302905

                 .673887

-2.44949            .214186

                 .953021

                 .214186

2                   - .707106

                 -2.4047E-7

                 .707107

MORE INPUT(1=YES, 0=NO)

? 0

END AT 6438

## §6 矩阵相加、相减和标量相乘

### 一、计算方法

标量与矩阵相乘

设标量=k, 矩阵为A

$$k \cdot A = k \cdot \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix} = \begin{pmatrix} ka_{11} & ka_{12} & \dots & ka_{1m} \\ ka_{21} & ka_{22} & \dots & ka_{2m} \\ \dots & \dots & \dots & \dots \\ ka_{n1} & ka_{n2} & \dots & ka_{nm} \end{pmatrix} \quad (14-6-1)$$

矩阵相加

$$A+B = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1m}+b_{1m} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2m}+b_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \dots & a_{nm}+b_{nm} \end{pmatrix} \quad (14-6-2)$$

矩阵相减

$$A-B = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix} - \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{pmatrix} = \begin{pmatrix} a_{11}-b_{11} & a_{12}-b_{12} & \dots & a_{1m}-b_{1m} \\ a_{21}-b_{21} & a_{22}-b_{22} & \dots & a_{2m}-b_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1}-b_{n1} & a_{n2}-b_{n2} & \dots & a_{nm}-b_{nm} \end{pmatrix} \quad (14-6-3)$$

### 二、程序说明

本程序所用的工作单元

A(N,M), S(M,N) 输入矩阵, 两个相加或相减的矩阵

R 用以计算的标量

Z 为选择运算种类变量

M,N为要计算矩阵的行数和列数, 开始时为所定的最大维数。

I,J,K为循环变量

### 三、上机操作说明

1. 由BASIC解释程序将本程序转入机内。
2. 用键盘命令RUN使本程序投入运行。电传打印出:  
"INPUT MATRIX DIMENSIONS N (LINE) AND M(COLUMN)." 根据要计算的矩阵中, 以元素最多的矩阵的维数进行回答, N是行数, M是列数。
3. 电传又打印出问话:  
"INPUT 1(SCALAR MULT.) OR 2(ADD) OR 3(SUBTRACT)" 这时用户可根据需要选择一种计算, 回答"1"矩阵与标量相乘, "2"两个矩阵相加, "3"两个矩阵相减。
4. 电传又打印出:  
"INPUT MATRIX A"  
这时用户首先输入矩阵A, 输入时按行输入, 每输入一个元素, 都要按回车键, 才能输入第二个元素。
5. 输入结束后, 电传又打印出:

"INPUT MATRIX B"

用户输入B矩阵, 输入方式同A矩阵

6. 输入完毕, 开始计算, 最后打印出结果

"OUTPUT RESULT MATRIX"

输出全部结果, 结果按列输出, 第一组数为第一列, 第二组为第二列, 等等。

7. 最后电传打印出

"MORE INPUT (1=YES, 0=NO) "

用户若要使本程序继续运行, 应回答1, 回答0时, 本程序运行结束。

#### 四、例题

$$a = 3.5$$

$$A = \begin{pmatrix} 1 & 2 & 3 & 3 \\ 2 & 3 & 4 & 4 \\ 4 & 5 & 6 & 8 \end{pmatrix}$$

$$a \times A = \begin{pmatrix} 3.5 & 7 & 10.5 & 10.5 \\ 7 & 10.5 & 14 & 14 \\ 14 & 17.5 & 21 & 28 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 2 & 3 & 3 \\ 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 8 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 8 & 9 & 11 \\ 5 & 2 & 7 & 3 \end{pmatrix}$$

$$A + B = \begin{pmatrix} 2 & 5 & 8 & 10 \\ 4 & 11 & 13 & 15 \\ 9 & 7 & 13 & 11 \end{pmatrix}$$

$$A - B = \begin{pmatrix} 0 & -1 & -2 & -4 \\ 0 & -5 & -5 & -7 \\ -1 & 3 & -1 & 5 \end{pmatrix}$$

#### 五、程序清单及运行结果

##### 程序清单

```
6500 REM PROGRAMMING OF MATRIX ADDING, MATRIX SUBTRACTING
6502 REM AND MULTIPLYING WITH SCALAR
6504 PRINT "INPUT MATRIX DIMENSIONS N(LINE) AND M(COLUMN)"
6506 INPUT N, M
6508 DIM A(N, M), B(N, M)
6510 PRINT
6512 PRINT "INPUT 1 (SCALAR MULT.) OR 2(ADD) OR 3(SUBTRACT)"
6514 INPUT Z
6516 PRINT
6518 IF Z > 1 THEN GOTO 6526
6520 PRINT "INPUT SCALAR"
6522 INPUT R
6524 PRINT
```

```

6526 PRINT "INPUT MATRIX A"
6528 FOR I=1 TO N
6530   FOR J=1 TO M
6532     INPUT A(I,J)
6534   NEXT J
6536   PRINT
6538 NEXT I
6540 IF Z=1 THEN GOTO 6566
6542 LET R=1
6544 PRINT "INPUT MATRIX B"
6546 FOR I=1 TO N
6548   PRINT
6550   FOR J=1 TO M
6552     INPUT B(I,J)
6554     IF Z=2 THEN GOTO 6558
6556     LET B(I,J)=-B(I,J)
6558     LET A(I,J)=A(I,J)+B(I,J)
6560   NEXT J
6562 NEXT I
6564 PRINT
6566 PRINT "OUTPUT RESULT MATRIX"
6568 FOR J=1 TO M
6570   PRINT
6572   FOR I=1 TO N
6574     PRINT R*A(I,J)
6576   NEXT J
6578   PRINT
6580 NEXT I
6582 PRINT "MORE INPUT (1=YES ,0=NO)"
6584 INPUT X
6586 PRINT
6588 IF X=1 THEN GOTO 6512
6590 END

```

程序运行结果:

```

RUN
INPUT MATRIX DIMENSIONS N(LINE)AND M(COLUMN)
? 3? 4
INPUT 1(SCALAR MULT.) OR 2(ADD) OR 3(SUBTRACT)
?
INPUT SALAR

```



? 3.5

INPUT MATRIX A

? 1? 2? 3? 3

? 2? 3? 4? 4

? 4? 5? 6? 8

OUTPUT RESULT MATRIX

3.5

7

14

7

10.5

17.5

10.5

14

21

10.5

14

28

MORE INPUT(1=YES,0=NO)

? 1

INPUT 1(SCALAR MULT.) OR 2(ADD) OR 3(SUBTRACT)

? 2

INPUT MATRIX A

? 1? 2? 3? 3

? 2? 3? 4? 4

? 4? 5? 6? 8

INPUT MATRIX B

? 1? 3? 5? 7

? 2? 8? 9? 11

? 5? 2? 7? 3

OUTPUT RESULT MATRIX

2

4

9

5

11

7

8

```

13
13
10
15
11
MORE INPUT (1=YES, 0=NO)
? 1
INPUT 1(SCALAR MULT.) OR 2(ADD) OR 3(SUBTRACT)
? 3
INPUT MATRIX A
? 1? 2? 3? 3
? 2? 3? 4? 4
? 4? 5? 6? 8
INPUT MATRIX B
? 1? 3? 5? 7
? 2? 8? 9? 11
? 5? 2? 7? 3
OUTPUT RESULT MATRIX
0
0
- 1
- 1
- 5
3
- 2
- 5
- 1
- 4
- 7
5
MORE INPUT (1=YES, 0=NO)
? 0
END AT 6590

```

## §7 矩阵求迹, 求秩

### 一、算法说明

对于任意矩阵 $A, a_{11}, a_{22}, \dots, a_{ii}$ 称为主对角线元素。定义主对角线元素之和为矩阵的迹。如果把主对角线元素以下的元素经过初等变换全部化为0, 那么, 主对角线上非零元素的个数即为该矩阵的秩。设有矩阵

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

1. 将第一行元素除以 $a_{11}$ ,

$$\text{即 } a_{1j}^{(1)} = a_{1j}/a_{11} \quad (j=1, 2, \cdots, n) \quad (14-7-1)$$

则原矩阵化为

$$\begin{pmatrix} 1 & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

2. 计算

$$a_{ij}^{(1)} = a_{ij} - a_{i1} a_{1j}^{(1)} \quad \left( \begin{matrix} i=2, 3, \cdots, m \\ j=1, 2, \cdots, n \end{matrix} \right) \quad (14-7-2)$$

则原矩阵化为

$$\begin{pmatrix} 1 & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \cdots & & & \\ 0 & a_{m2}^{(1)} & \cdots & a_{mn}^{(1)} \end{pmatrix}$$

3. 将第二行元素除以 $a_{22}^{(1)}$ , 即:

$$a_{2j}^{(2)} = a_{2j}^{(1)} / a_{22}^{(1)} \quad (j=2, 3, \cdots, n)$$

4. 再作计算:

$$a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i2}^{(1)} \cdot a_{2j}^{(2)} \quad \left( \begin{matrix} i=3, 4, \cdots, m \\ j=2, 3, \cdots, n \end{matrix} \right) \quad (14-7-3)$$

则原矩阵化为

$$\begin{pmatrix} 1 & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & 1 & \cdots & a_{2n}^{(2)} \\ 0 & 0 & \cdots & a_{3n}^{(2)} \\ \cdots & & & \\ 0 & 0 & \cdots & a_{mn}^{(2)} \end{pmatrix}$$

如此一直做下去, 直到主对角线以下元素全部化为零。然后确定主对角线上非零元素的个数, 即得矩阵的秩。

## 二、程序说明

程序使用变量及工作单元

M, N 矩阵的行、列数

A(M, N) 存放输入矩阵。运行结束后, A 中信息破坏。

工作单元: S, I, J, K, B, R, T, Y

## 三、上机操作步骤

1. 用BASIC解释程序将源程序送入机内, 发键盘命令 RUN ↵ 使本程序投入运行, 电传自动印出:

"INPUT THE ROW AND COLUMN OF THE MATRIX"

要求回答矩阵的行、列数。这时用户应打入两个数，两个数之后都要按回车键。

2. 回答后，又印出：

"INPUT THE MATRIX IN ORDER OF ROW"

这时用户应按行将矩阵每个元素输入，每输入一个数都要按回车键。

3. 数据全部送入后，机器开始运行，印出计算结果（B为矩阵的迹，R为秩）

"THE LOCUS OF THE MATRIX IS B="

"THE RANK OF THE MATRIX IS R="

接着，又打印出：

ANY MORE, PLEASE TYPE IN 1 FOR YES OR IN 0 FOR NO

问用户是否还要计算，若是，则回答1，否则回答0。

4. 回答1后，程序从头重新开始，回答0时，运行结束。

#### 四、例题

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 1 & 3 & 5 & 7 \\ 1 & 3 & 13 & 17 \end{pmatrix}$$

矩阵的秩 = 3

矩阵的迹 = 27

#### 五、程序清单及运行结果

##### 程序清单

```
6600 PRINT "INPUT THE ROW AND COLUMN OF THE MATRIX",
6602 INPUT M,N
6604 PRINT
6606 DIM A(M,N)
6608 PRINT "INPUT THE MATRIX IN ORDER OF ROW"
6610 FOR I=1 TO M
6612   FOR J=1 TO N
6614     INPUT A(I,J)
6616   NEXT J
6618   PRINT
6620 NEXT I
6622 LET K=M
6624 IF M=<N THEN GOTO 6628
6626 LET K=N
6628 LET B=0
6630 FOR I=1 TO K
6632   LET B=B+A(I,I)
6634 NEXT I
6636 PRINT "THE LOCUS OF THE MATRIX IS B=";B
6638 LET S=1
6640 FOR I=S TO N
```

```

6642  IF I>M THEN GOTO 6682
6644  FOR J=1 TO M
6646      IF A(J,I)><0 THEN GOTO 6652
6648  NEXT J
6650  GOTO 6708
6652  IF J=1 THEN GOTO 6664
6654  FOR K=1 TO N
6656      LET T=A(I,K)
6658      LET A(I,K)=A(J,K)
6660      LET A(J,K)=T
6662  NEXT K
6664  IF I=M THEN GOTO 6682
6666  LET T=1/A(I,I)
6668  FOR K=I+1 TO N
6670      LET A(I,K)=A(I,K)*T
6672      FOR L=I+1 TO M
6674          LET A(L,K)=A(L,K)-A(I,K)*A(L,I)
6676      NEXT L
6678  NEXT K
6680  NEXT I
6682  LET R=0
6684  IF M=<N THEN GOTO 6688
6686  LET M=N
6688  FOR I=1 TO M
6690      IF A(I,I)=0 THEN GOTO 6696
6692      LET R=R+1
6694  NEXT I
6696  PRINT "THE RANK OF THE MATRIX R=";R
6698  PRINT "ANY MORE? PLEASE TYPE IN 1 FOR YES OR IN 0 FOR
      NO";
6700  INPUT Y
6702  PRINT
6704  IF Y=1 THEN GOTO 6608
6706  END
6708  LET S=I
6710  FOR K=1 TO M
6712      LET A(K,S)=A(K,N)
6714  NEXT K
6716  LET N=N-1
6718  IF N<S THEN GOTO 6682

```

6720 GOTO 6640

程序运行结果:

RUN

INFUT THE ROW AND COLUMN OF THE MATRIX ? 4 ? 4

INPUT THE MATRIX IN ORDER OF ROW

? 1 ? 2 ? 3 ? 4

? 2 ? 4 ? 6 ? 8

? 1 ? 3 ? 5 ? 7

? 1 ? 3 ? 13 ? 17

THE LOCUS OF THE MATRIX IS B= 27

THE RANK OF THE MATRIX B= 3

ANY MORE? PLEASE TYPE IN 1 FOR YES OR IN 0 FOR NO ? 0

END AT 6706

## §8 迭代法解线性方程组

### 一、方法概要

对于线性方程组

$$AX=B$$

(14-8-1)

我们以 $n=3$ 为例来说明迭代法的基本原理。将(14-8-1)式写成

$$\begin{cases} a_{11}x_1 = -a_{12}x_2 - a_{13}x_3 + b_1 \\ a_{21}x_2 = -a_{21}x_1 - a_{23}x_3 + b_2 \\ a_{31}x_3 = -a_{31}x_1 - a_{32}x_2 + b_3 \end{cases} \quad (14-8-2)$$

取任一向量 $x^{(0)} = (x_1^{(0)} \ x_2^{(0)} \ x_3^{(0)})^T$ 作为初始预估解向量,代入(14-8-2)式右端算出一个新的向量 $x^{(1)} = (x_1^{(1)} \ x_2^{(1)} \ x_3^{(1)})^T$ 。可以看出,应用这种方法时,应当要求 $a_{11} \neq 0$ ,  $a_{22} \neq 0$ ,  $a_{33} \neq 0$ 。再把 $x^{(1)}$ 代入(14-8-2)式右端可求得 $x^{(2)} = (x_1^{(2)} \ x_2^{(2)} \ x_3^{(2)})^T$ ,这样继续做下去就可得到一个序列

$$x^{(0)}, x^{(1)}, \dots, x^{(m)}, x^{(m+1)}, \dots$$

其第 $m+2$ 项 $x^{(m+1)}$ 是由前项 $x^{(m)}$ 按下式求得的

$$\begin{cases} a_{11}x_1^{(m+1)} = -a_{12}x_2^{(m)} - a_{13}x_3^{(m)} + b_1 \\ a_{21}x_2^{(m+1)} = -a_{21}x_1^{(m)} - a_{23}x_3^{(m)} + b_2 \\ a_{31}x_3^{(m+1)} = -a_{31}x_1^{(m)} - a_{32}x_2^{(m)} + b_3 \end{cases} \quad (14-8-3)$$

可以证明,当系数矩阵满足一定条件时,序列趋向于方程组的精确解。

推广到 $n$ 阶,我们可以得出一般的迭代公式

$$\begin{aligned} x_i^{(m+1)} &= \frac{1}{a_{ii}} \left\{ b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m)} - \sum_{j=i+1}^n a_{ij}x_j^{(m)} \right\} \\ &= x_i^{(m)} + \frac{1}{a_{ii}} \left\{ b_i - \sum_{j=1}^n a_{ij}x_j^{(m)} \right\} \\ &\quad (i=1, 2, \dots, n, \quad m=1, 2, \dots) \end{aligned} \quad (14-8-4)$$

任意给定  $X_i^{(0)}$  ( $i=1,2,\dots,n$ ), 按照 (14—8—4) 式迭代若干次 (假如收敛的话) 则可求得满足一定精度要求的一组近似解。

## 二、程序说明

程序使用变量及工作单元

N 方程组未知数个数。

A(N,N+1) 存放增广系数矩阵

X(N) 存放每次迭代的解向量

S, B 运算过程中的中间变量

M 最大迭代次数

D 允许的绝对误差

I, J, T, Y 为工作单元

## 三、上机操作说明

1. 用BASIC解释程序将本程序送入机内, 键盘命令RUN↵, 则电传机自动印出,

"INPUT THE NUMBER OF UNKNOWNs?"

要求回答未知数的个数, 这时用户应打入未知数的个数, 并按回车键。

2. 回答后, 又打印出:

"INPUT THE MAX ITERATIONS AND ALLOWED ERROR

这时用户应回答最大迭代次数及允许误差, 每回答一个数后都要按回车键。

3. 回答后, 又打印出:

"INPUT THE AUGMENTED COEFFICIENT MATRIX"

这时用户应按行输入方程未知数的系数及常数项。每行输入N+1个数 每个数后都要回车。

4. 输入完毕后, 机器自动运行。如果迭代次数在M以内就已经满足精度要求, 则打印出结果:

X(1)= \* \* \*

X(2)= \* \* \*

.....

X(n)= \* \* \*

若迭代M次后仍不满足精度要求, 则打印出:

"CONVERGENCE TOO SLOW. LAST VALUES COMPUTED ARE:"

X(1)= \* \* \*

X(2)= \* \* \*

.....

X(n)= \* \* \*

这时打出的结果为最后一次迭代所求得的值。

在上述两种情况下, 都将接着打印出:

"ANY MORE ? INPUT 1 FOR YES OR 0 FOR NO ?"

若要继续运算, 则回答1, 否则回答0, 回答1后, 程序从头重新开始。回答0后, 程序运行结束。

## 四、例题

$$\begin{cases} 12x_1 - 3x_2 + x_3 + 4x_4 = 25 \\ x_1 + 15x_2 - 7x_3 - x_4 = 6 \\ 4x_1 + x_2 - 20x_3 + 5x_4 = -34 \\ 2x_1 - 8x_2 + x_3 + 10x_4 = 29 \end{cases}$$

计算结果:

$$\begin{cases} x_1 = 1.01276 \\ x_2 = 1.88152 \\ x_3 = 3.01802 \\ x_4 = 3.90086 \end{cases}$$

## 五、程序清单及运行结果

### 程序清单

```

6710 PRINT "INPUT THE NUMBER OF UNKNOWN",
6712 INPUT N
6714 PRINT
6716 PRINT "INPUT THE MAX ITERATIONS AND ALLOW ERROR",
6718 INPUT M,D
6720 PRINT
6722 DIM A(N,N+1),X(N)
6724 PRINT "INPUT THE AUGMENTED COEFFICIENT MATRIX"
6726 FOR I=1 TO N
6728     FOR J=1 TO N+1
6730         INPUT A(I,J)
6732     NEXT J
6734     PRINT
6736     LET X(I)=A(I,N+1)
6738 NEXT I
6740 FOR K=1 TO M
6742     LET T=0
6744     FOR I=1 TO N
6746         LET S=0
6748         FOR J=1 TO N
6750             LET S=S+A(I,J)*X(J)
6752         NEXT J
6754         LET B=X(I)
6756         LET X(I)=B+(A(I,N+1)-S)/A(I,I)
6758         IF ABS(X(I)-B)=<D THEN GOTO 6762
6760         LET T=T+1
6762     NEXT I
6764     IF T=0 THEN GOTO 6770
6766 NEXT K

```



```

6768 PRINT "CONVERENCE TOO SLOW. LAST VALUES COMPUTED ARE";
6770 FOR I=1 TON
6772     PRINT "X(";I;")="; X(I)
6774 NEXT I
6776 PRINT "ANY MORE? INPUT 1 FORYES, OR 0 FOR NO";
6778 INPUT Y
6780 PRINT
6782 IF Y=1 THEN GOTO 6710
6784 END

```

程序运行结果:

RUN

INPUT THE NUMBER OF UNKNOWNNS ? 4

INPUT THE MAX ITERATIONS AND ALOW ERROR ? 50 ? .999

INPUT THE AUGMENTED COEFFICIENT MATRIX

```

? 12 ? -3 ? 1 ? 4 ? 25
? 1 ? 15 ? -7 ? -1 ? 6
? 4 ? 1 ? -20 ? 5 ? -34
? 2 ? -8 ? 1 ? 10 ? 29

```

X(1)=1.01276

X(2)=1.88152

X(3)=3.01802

X(4)=3.90086

ANY MORE? INPUT 1 FOR YES OR 0 FOR NO ? 0

END AT 6784

## §9 LU分解法解线性方程组

### 一、算法说明

设有方程组

$$AX=B$$

(14-9-1)

其系数矩阵A可以分解成下三角阵L与上三角阵U的乘积, 即  $A=LU$

则  $LUX=B$

其中

$$L = \begin{pmatrix} 1_{11} & & & \\ l_{21} & 1_{22} & & \\ \vdots & \vdots & \ddots & \\ \vdots & \vdots & & \\ l_{n1} & l_{n2} & \dots & 1_{nn} \end{pmatrix}$$

$$U = \begin{pmatrix} 1 & u_{12} & \dots & u_{1n} \\ & 1 & \dots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{n-1,n} \\ & & & & 1 \end{pmatrix}$$

令  $UX=Y$ , 则方程 (14-9-1) 可分解为两个方程组

$$\begin{cases} LY=B \\ UX=Y \end{cases} \quad \begin{matrix} (14-9-2) \\ (14-9-3) \end{matrix}$$

先从 (14-9-2) 式解出  $Y$ , 再由 (14-9-3) 式解出  $X$

如何根据  $A$  求得  $L$  和  $U$  呢?

因为  $LU=A$  即

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & \dots & \dots & l_{nn} \end{pmatrix} \cdot \begin{pmatrix} 1 & u_{12} & \dots & u_{1n} \\ & 1 & \dots & u_{2n} \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

根据矩阵乘法规则, 可以看出, 左边乘积的第一列就等于  $l_{11}, l_{21}, \dots, l_{n1}$ , 所以  $l_{i1}=a_{i1}$  ( $i=1, 2, \dots, n$ ), 而左边乘积的第一行为  $l_{11}, l_{11}u_{1j}$  ( $j=2, 3, \dots, n$ ), 与右边相比, 得:

$$u_{1j} = a_{1j}/l_{11} = a_{1j}/a_{11} \quad (j=2, 3, \dots, n) \quad (14-9-4)$$

以  $L$  的第  $i$  行乘以  $U$  的第  $j$  列 ( $i \geq j$ ) 得:

$$l_{i1}u_{1j} + l_{i2}u_{2j} + \dots + l_{i,i-1}u_{i-1,j} + l_{ii} = a_{ij} \quad (14-9-5)$$

$$\text{所以有 } l_{ii} = a_{ii} - \sum_{k=1}^{i-1} l_{ik}u_{kj} \quad (i \geq j)$$

以  $L$  的第  $i$  行乘  $u$  的第  $j$  列 ( $i < j$ ) 得:

$$l_{i1}u_{1j} + l_{i2}u_{2j} + \dots + l_{ii}u_{ij} = a_{ij}$$

$$\text{所以有 } u_{ij} = 1/l_{ii} (a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}) \quad (i < j) \quad (14-9-6)$$

这样就可求出  $L$  和  $U$  的全部元素。

可以看出, 矩阵  $L$  和  $U$  的值只与  $A$  的值有关, 而与  $B$  无关, 因此, 重复解多次  $B$  变化而  $A$  不变的方程组时, 只须进行一次分析即可, 而每次只须将  $B$  代入 (14-9-2), (14-9-3) 式即可求得方程组的解。所以这种方法特别适宜于求解  $A$  不变而  $B$  多次变化的方程组, 可大大节省计算时间。

## 二、程序说明

程序使用数组  $A(N, N+1)$ , 前几列存放系数矩阵, 分解后存放  $(L+U-I)$ , 最后一列存放方程排列序号。

$X(N)$  存放  $B$ , 运行结束后存放解。

本程序所用工作单元:  $L, I, J, Y, B, N$

## 三、上机操作步骤:

1. 由 BASIC 解释程序将源程序送入机内, 键盘命令  $RUN \swarrow$ , 电传打印出:  
"INPUT THE NUMBER OF UNKNOWNNS" 要求输入未知数的个数,
2. 以后又打印出:  
"INPUT THE COEFFICIENT MATRIX" 这时按行输入系数矩阵,
3. 输入后, 程序开始进行  $LU$  分解, 若矩阵非奇异, 则打印出;

"IN FOLLOWING SEQUENCE INPUT THE CONSTANT TERMS"

$B(i_1)=?$

这时应输入第 $i_1$ 个方程的常数项 ( $i_1$ 是1到N中的某个数)。

直到将N个方程的常数项全部输入后, 程序开始运算, 并打印出计算结果。若矩阵奇异则打印出: "NO UNIQUE SOLUTION"

在以上两种情况下, 最后打印出:

"ANY MORE? INPUT 1 OR 2 FOR YES, 0 FOR NO"

若用户还要计算, 并且系数矩阵也要变, 则回答2; 若只改变B, 则回答1; 若计算结束, 则回答0。

#### 四、例题

$$\begin{cases} 3X_1 + 2X_2 + 4X_3 + X_4 = 5 \\ 2X_1 + \quad + 2X_3 + 5X_4 = 1 \\ 2X_1 + X_2 + 2X_3 + X_4 = 3 \\ 2X_1 + 4X_2 + 3X_3 + X_4 = 4 \end{cases}$$

其解为

$$\begin{cases} X_1 = 1.42857 \\ X_2 = 0.285714 \\ X_3 = 0.142858 \\ X_4 = -0.428572 \end{cases}$$

#### 五、程序清单及运行结果

##### 程序清单

```
6790 PRINT "INPUT THE NUMBER OF UNKNOWN";
6792 INPUT N
6794 PRINT
6796 DIM A(N,N+1),X(N)
6798 PRINT "INPUT THE COEFFICIENT MATRIX"
6800 FOR I=1 TO N
6802   FOR J=1 TO N
6804     INPUT A(I,J)
6806   NEXT J
6808   PRINT
6810   LET A(I,N+1)=I
6812 NEXT I
6814 GOSUB 6843
6816 PRINT "IN FOLLOWING SEQUENCE INPUT THE CONSTANT TERMS"
6818 FOR I=1 TO N
6820   PRINT "B(",A(1,N+1);")=";
6822   INPUT X(I)
6824   PRINT
```

```

6826 NEXT I
6828 GOSUB 6848 6848 6848
6830 FOR I=1 TO N
6832 PRINT "X(";I;")=";X(I)
6834 NEXT I
6836 PRINT "ANY MORE? INPUT 1 OR 2 FOR YES, 0 FOR NO";
6838 INPUT Y
6840 PRINT
6842 IF Y=1 THEN GOTO 6816
6844 IF Y=2 THEN GOTO 6790
6846 END
6848 FOR J=1 TO N-1
6850 LET L=J
6852 LET B=ABS(A(J,J))
6854 FOR I=J+1 TO N
6856 IF ABS(A(I,J))=<B THEN GOTO 6862
6858 LET L=I
6860 LET B=ABS(A(I,J))
6862 NEXT I
6864 IF B=0 THEN GOTO 6894
6866 IF L=J THEN GOTO 6878
6868 FOR K=1 TO N+1
6870 LET B=A(L,K)
6872 LET A(L,K)=A(J,K)
6874 LET A(J,K)=B
6876 NEXT K
6878 FOR K=J+1 TO N
6880 LET A(J,K)=A(J,K)/A(J,J)
6882 FOR I=J+1 TO N
6884 LET A(I,K)=A(I,K)-A(I,J)*A(J,K)
6886 NEXT I
6888 NEXT K
6890 NEXT J
6892 RETURN
6894 PRINT "NO UNIQUE SOLUTION"
6896 GOTO 6836
6898 LET X(1)=X(1)/A(1,1)
6900 FOR I=2 TO N
6902 FOR J=1 TO I-1
6904 LET X(I)=X(I)-A(I,J)*X(J)

```

```

6906 NEXT J
6908 LET X(I)=X(I)/A(I,I)
6910 NEXT I
6912 FOR I=N-1 TO 1 STEP -1
6914   FOR J=N TO I+1 STEP -1
6916     LET X(I)=X(I)-A(I,J)*X(J)
6918   NEXT J
6920 NEXT I
6922 RETURN

```

程序运行结果:

RUN

INPUT THE NUMBER OF UNKNOWNNS ? 4

INPUT THE COEFFICIENT MATRIX

? 3 ? 2 ? 4 ? 1

? 2 ? 0 ? 2 ? 5

? 2 ? 1 ? 2 ? 1

? 2 ? 4 ? 3 ? 1

IN FOLLOWING SEQUENCE INPUT THE CONSTANT TERMS

B(1)=? 5

B(4)=? 4

B(3)=? 3

B(2)=? 1

X(1)= 1.42857

X(2)= .285714

X(3)= .142858

X(4)=-.428572

ANY MORE? INPUT 1 OR 2 FOR YES, 0 FOR NO ? 0

END AT 6846

## §10 高斯——约旦法解线性方程组

### 一、计算方法

设有方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \quad (14-10-1)$$

写成矩阵形式

$$AX=B$$

其中, A为系数矩阵, 即:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

X为列向量, 即  $X = (x_1, x_2, \dots, x_n)^T$

B为列向量, 即  $B = (b_1, b_2, \dots, b_n)^T$

把A、B合并为增广矩阵  $[A \vdots B]$ , 对增广矩阵进行行初等变换, 使A化为单位矩阵, 同时B也化为C, 即得  $IX=C$ , 所以C即解向量X,

对增广矩阵进行初等变换, 可分下列两个步骤进行。

(1) 归1。用式子表示即

$$a_{ij}^{(1)} = a_{ij}^{(0)} / a_{ii}^{(0)} \quad (j=1, 2, \dots, n)$$

$$b_i^{(1)} = b_i^{(0)} / a_{ii}^{(0)}$$

式中右上角的(0)表示原来的元素, (1)表示作第一次变换后的元素。

(2) 消元。

用第i行 ( $i=2, 3, \dots, n$ ) 元素减去第一行相应元素的 $a_{i1}$ 倍, 使 $a_{i1}$ 变为0, 用式子表示, 即

$$a_{ij}^{(1)} = a_{ij}^{(0)} - a_{i1}^{(0)} a_{1j}^{(1)} \quad (i=2, 3, \dots, n, j=1, 2, \dots, n) \quad (14-10-2)$$

$$b_i^{(1)} = b_i^{(0)} - a_{i1}^{(0)} b_1^{(1)} \quad (i=2, 3, \dots, n) \quad (14-10-3)$$

于是增广矩阵由  $[A^{(0)} \vdots B^{(0)}]$  化为  $[A^{(1)} \vdots B^{(1)}]$ , 这是第一次变换。

同理, 再进行第二次、第三次、...变换, 经过几次变换增广矩阵化为

$$[A^{(n)} \vdots B^{(n)}] = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & b_1^{(n)} \\ 0 & 1 & 0 & \dots & 0 & b_2^{(n)} \\ 0 & 0 & 1 & \dots & 0 & b_3^{(n)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & b_n^{(n)} \end{pmatrix}$$

这时方程组的解已经求出, 即

$$X = B^{(n)}$$

从以上分析过程可以看到, 在进行除以 $a_{ii}$ 的运算时, 如果遇到 $a_{ii}=0$ , 则运算将会中断。即使 $a_{ii} \neq 0$ , 如果 $a_{ii}$ 较小时, 也会引起较大误差, 为了避免这种情况, 可采用所谓选主元的办法, 即在每次归1前, 先在相应的列中找出绝对值最大的元素, 通过行与行交换, 将该元素换到主对角线的位置, 然后再进行归1, 这种选主元的办法称为列主元法, 还有所谓全主元法, 本程序采用的是列主元法。

## 二、程序说明

程序使用数组A(N, N+1)存放增广系数矩阵(N为未知数个数), 运行结果, 原信息全部破坏, 最后一列为所求的解向量。

本程序所用的工作单元: I, J, S, T, C, B, X, N

## 三、上机操作说明

1. 用BASIC解释程序将本程序送入机内, 键盘命令RUN启动本程序, 电传机打印

出: "NUMBER OF UNKNOWNNS N" 这时用户应回答未知数的个数。

2. 回答后, 又打印出:

"INPUT THE AUGMENTED COEFFICIENT MATRIX", 这时用户应按行打入增广系数矩阵, 每行  $N+1$  个数, 每个数后都要回车

3. 输入完毕后, 机器自动运行。如果系数矩阵非奇异, 则打出唯一解

$X(1)=***$

$X(2)=***$

.....

$X(n)=***$

如果系数矩阵奇异, 则打出:

"NO UNIQUE SOLUTION"

在上述两种情况下, 都接着打印出:

"ANY MORE? INPUT 1 FOR YES OR 0 FOR NO?" 用户若还要运算, 则回

答1, 否则回答0。回答1后, 程序从头开始运行, 回答0后, 结束。

#### 四、例题

$$\begin{cases} 12x_1 - 3x_2 + x_3 + 4x_4 = 25 \\ x_1 + 15x_2 - 7x_3 - x_4 = 6 \\ 4x_1 + x_2 - 20x_3 + 5x_4 = -34 \\ 2x_1 - 8x_2 + x_3 + 10x_4 = 29 \end{cases}$$

计算结果

$$\begin{cases} x_1 = 1.01276 \\ x_2 = 1.88152 \\ x_3 = 3.01802 \\ x_4 = 3.90086 \end{cases}$$

计算结果与运行结果之间误差较大, 是由于本计算方法变换引起。

#### 五、程序清单及运行结果

##### 程序清单

```
7850 PRINT "NUMBER OF UNKNOWNNS N"
7852 INPUT N
7854 PRINT
7856 DIM A(N,N+1)
7858 PRINT "INPUT THE AUGMENTED COEFFICIENT MATRIX"
7860 FOR I=1 TO N
7862   FOR J=1 TO N+1
7864     INPUT A(I,J)
7866   NEXT J
7868 PRINT
7870 NEXT I
7872 PRINT
7874 FOR S=1 TO N
```

```

7876  FOR T=S TO N
7878      IF A(T,S)><0 THEN GOTO 7886
7880      NEXT T
7882      PRINT "NO UNIQUE SOLUTION"
7884      GOTO 7948
7886      GOSUB 7908
7888      LET C=1/A(S,S)
7890      GOSUB 7918
7892      FOR T=1 TO N
7894          IF T=S THEN GOTO 7900
7896          LET C=-A(T,S)
7898          GOSUB 7926
7900      NEXT T
7902  NEXT S
7904  GOTO 7934
7906  FOR J=1 TO N+1
7908      LET B=A(S,J)
7910      LET A(S,J)=A(T,J)
7912      LET A(T,J)=B
7914  NEXT J
7916  RETURN
7918  FOR J=1 TO N+1
7920      LET A(S,J)=C * A(S,J)
7922  NEXT J
7924  RETURN
7926  FOR J=1 TO N+1
7928      LET A(T,J)=A(T,J)+C * A(S,J)
7930  NEXT J
7932  RETURN
7934  FOR T=1 TO N
7936      PRINT "X(",T;")=",A(T,N+1)
7938  NEXT T
7940  PRINT "ANY MORE? INPUT 1 FOR YES OR 0 FOR NO"
7942  INPUT X
7944  PRINT
7946  IF X=1 THEN GOTO 7858
7948  END

```

程序运行结果:

RUN

NUMBER OF UNKNOWN N



```

? 4
INPUT THE AUGMENTED COEFFICIENT MATRIX
? 12 ? -3 ? 1 ? 4 ? 25
? 1 ? 15? - 7 ? -1 ? 6
? 4 ? 1 ? -20 ? 5 ? -34
? 2 ? -8 ? 1 ? 10 ? 29
X(1)= 1
X(2)= 2
X(3)= 3
X(4)= 4
ANY MORE? INPUT 1 FOR YES OR 0 FOR NO
? 0
END AT 7948

```

## §11 化一般矩阵为上H阵

### 一、方法概要

定理1: 若A是分块三角形矩阵, 并且对角块都是方阵, 则它的全部特征值由各对角块的特征值组成。

设: A如(\*)所示的分块三角形矩阵, 则其特征值为:

$$2, 2, 3, 1 \pm j$$

$$A = \begin{pmatrix} 2 & 9 & 6 & -5 & 7 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & -2 & 4 & 3 & 2 \\ 0 & 5 & 7 & 8 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \quad (*)$$

定理2: 相似矩阵的特征值相同, 或矩阵的相似变换保持特征值不变。

根据前述两个定理, 任给一个矩阵  $A = A_1$ , 对它连续进行相似变换得:

$$A_2 = C_1^{-1} A_1 C_1, A_3 = C_2^{-1} A_2 C_2, \dots, A_{k+1} = C_k^{-1} A_k C_k, \dots$$

使所得的矩阵序列  $\{A_k\}$  收敛为分块三角形矩阵, 并且对角块都是  $1 \times 1$  或  $2 \times 2$  阶矩阵, 那么任意一个矩阵A的全部特征值都可以求出来。

以上就是用QR法求矩阵的全部特征值的基本原理, 由于此法是首先将A阵化为上H阵, 故要对上H阵作一些介绍:

上H阵, 又称准三角阵 (Hessenberg型矩阵) 如下所示:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \cdots & h_{1n} \\ h_{21} & h_{22} & h_{23} \cdots & h_{2n} \\ & h_{32} & h_{33} \cdots & h_{3n} \\ & & \ddots & \vdots \\ & & & h_{n, n-1} & h_{nn} \end{pmatrix}$$

若把元素 $h_{ii}$ 所在的对角线称为主对角线,  $h_{i, i-1}$ 所在的对角线称为次对角线, 则准三角矩阵就是次对角线以下的元素 $h_{ij}$  ( $j < i-1$ ) 全等于零的矩阵。

定义: 如果准三角阵的次对角线元素都不等于零, 则称之为是不可约的; 反之, 如果次对角线有另元素, 则称之为可约的。

例:

$$A = \begin{pmatrix} 1 & 5 & 8 & 10 \\ 5 & 2 & 6 & 4 \\ 0 & 2 & 3 & 7 \\ 0 & 0 & 9 & 4 \end{pmatrix} \quad \text{是不可约准三角阵。}$$

$$B = \begin{pmatrix} 3 & -1 & 0 & 2 \\ 4 & 6 & -7 & 9 \\ 0 & 0 & 5 & -8 \\ 0 & 0 & 3 & 1 \end{pmatrix} \quad \text{是可约准三角阵。}$$

准三角形矩阵对QR算法有两个有利的地方:

第一, 可约准三角阵可分割成分块准三角阵, 因此对于准三角阵的QR算法, 可归结为对不可约准三角阵的算法。

第二, QR变换保持准三角阵不变。

化一般矩阵为上H阵有两种方法, 一种是用初等相似变换, 另一种是用初等对称正交变换, 即豪氏变换。本程序是用初等相似变换。

用初等相似变换把一矩阵A化为上H阵, 需要通过 $n-2$ 个主要步骤, 它们依次把矩阵A中的第1, 2, ...,  $n-2$ 列化为上H阵。

现假设 $A_r$ 为A经过 $r-1$ 步变换所得的矩阵, 对于实矩阵, 其第 $r$ 步变换的计算方法如下:

(1) 选  $\left| a_{(r+1), r}^{(r)} \right|$  ( $i=r+1, \dots, n$ ) 中最大值 (若不止一个, 取先遇到的第一个), 并用  $\left| a_{(r+1), r}^{(r)} \right|$  表示 (称为主元)。

(2) 交换第 $(r+1)'$ 行与第 $(r+1)$ 行, 交换第 $(r+1)'$ 列与第 $(r+1)$ 列。

(3) 对于从 $r+2$ 到 $n$ 的每一个 $i$ 值, 计算如下:

$$n_{i, r+1} = a_{i, r}^{(r)} / a_{r+1, r}^{(r)} \quad (14-11-1)$$

$$a_{ij} \leftarrow a_{ij} - n_{i, r+1} \times a_{r+1, j} \quad (j=1, 2, \dots, n) \quad (14-11-2)$$

$$a_{i, r+1} \leftarrow a_{i, r+1} + n_{i, r+1} \times a_{ii} \quad (j=1, 2, \dots, n) \quad (14-11-3)$$

(如果 $n_{i, r+1}=0$ , 则(2)和(3)不需计算)。

根据上面的计算, 从 $A_r$ 到 $A_{r+1}$ , 可用下面的关系表示:

$$A_{r+1} = N_{r+1}^{-1} I_{r+1, (r+1)'} A_r I_{r+1, (r+1)'} N_{r+1}$$

这里矩阵 $I_{r+1, (r+1)'}$ 是初等置换矩阵。(即交换行或列的变换矩阵) 而 $N_{r+1}$ 是一初等阵, 即

$$(N_{r+1})_{i, r+1} = n_{i, r+1} \quad (i=r+2, \dots, n)$$

其余元素与单位矩阵相同, 即

$$(N_{r+1})_{ij} = \delta_{ij}$$

## 二、程序说明

1. 使用数组A, B。
2. 使用变量I, J, L, M, X, Y。
3. 程序结束, 原来的信息破坏, 求出的H阵在A中(简化过程中的数 $a_{i,r+1}$ 也存放在A的(i, r)位置中, 若需要将其打印出来, 则需删去语句7106, 7108和7110)。B记录简化过程中行列的置换次数。

## 三、上机操作使用

1. 由BASIC解释程序引入本程序, 键盘命令RUN启动本程序执行, 机器自动印出:  
“INPUT DEGREE N OF MATRIX A” 询问给定矩阵的维数。
2. 回答矩阵A的维数, 机器又自动印出:  
“INPUT ELEM. OF MATRIX A” 要求输入矩阵A的元素。
3. 按行输入矩阵A的元素之后, 机器立即可以将结果矩阵印出来。

## 四、试题

$$A = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & 4 & 1 & 6 \\ 1 & 2 & -1 & 3 \\ 2 & 0 & 1 & 3 \end{pmatrix}$$

结果为:

$$H = \begin{pmatrix} 1 & 8.5 & 5.3214287 & 3 \\ 2 & 10.5 & 6.107142156 & 1 \\ 0 & -7 & -3 & 0 \\ 0 & 0 & 0.1607142856 & -1.5 \end{pmatrix}$$

## 五、程序清单及运行结果

### 程序清单

```
7000 REM CLMHES PROCEDURE
7002 PRINT "INPUT DEGREE N OF MATRIX A";
7004 INPUT N
7006 PRINT
7008 PRINT "INPUT ELEM. OF MATRIX A"
7010 DIM A(N,N+4),B(N)
7012 FOR I=1 TO N
7014   FOR J=1 TO N
7016     INPUT A(I,J)
7018   NEXT J
7020 PRINT
7022 LET B(I)=I
7024 NEXT I
7026 LET L=N-1
7028 FOR M=2 TO L
7030 LET I=M
```

```

7032 LET X=0
7034 FOR J=M TO N
7036     IF ABS(A(J,M-1))=<ABS(X) THEN GOTO 7042
7038     LET X=A(J,M-1)
7040     LET I=J
7042 NEXT J
7044 IF I=M THEN GOTO 7072
7046 LET Y=B(M)
7048 LET B(M)=B(I)
7050 LET B(I)=Y
7052 FOR J=M-1 TO N
7054     LET Y=A(I,J)
7056     LET A(I,J)=A(M,J)
7058     LET A(M,J)=Y
7060 NEXT J
7062 FOR J=1 TO N
7064     LET Y=A(J,I)
7066     LET A(J,I)=A(J,M)
7068     LET A(J,M)=Y
7070 NEXT J
7072 IF X=0 THEN GOTO 7098
7074 FOR I=M+1 TO N
7076     LET Y=A(I,M-1)
7078     IF Y=0 THEN GOTO 7096
7080     LET Y=Y/X
7082     LET A(I,M-1)=Y
7084     FOR J=M TO N
7086         LET A(I,J)=A(I,J)-Y*A(M,J)
7088     NEXT J
7090     FOR J=1 TO N
7092         LET A(J,M)=A(J,M)+Y*A(J,I)
7094     NEXT J
7096 NEXT I
7098 NEXT M
7100 PRINT "OUTPUT THE UPPER H MATRIX",TAB(50);"ROW NUMBER"
7102 FOR I=1 TO N
7104     FOR J=1 TO N
7106         IF I=<2 THEN GOTO 7112
7108         IF J>I-2 THEN GOTO 7112
7110         LET A(I,J)=0

```

```

7112     PRINT A(I,J);
7114     NEXT J
7116     PRINT TAB(55);B(I)
7118     PRINT
7120     NEXT I
7122     END

```

程序运行结果:

RUN

INPUT DEGREE N OF MATRIX A? 4

INPUT ELEM.OF MATRIX A

```

? 1 ? 2 ? 3 ? 5
? 2 ? 4 ? 1 ? 6
? 1 ? 2 ? -1 ? 3
? 2 ? 0 ? 1 ? 3

```

OUTPUT THE UPPER H MATRIX

ROW NUMBER

1	8.5	5.32143	3	1
2	10.5	6.10714	1	2
0	-7	-3	0	4
0	0	.160714	-1.5	3

END AT 7122

## §12 QR法求实上H阵特征值

### 一、方法概要

如上节所述对于任意给定的矩阵 $A=A_1$ , 如果对它连续进行相似变换:

$$A_2 = C_1^{-1} A_1 C_1, A_3 = C_2^{-1} A_2 C_2, \dots, A_{k+1} = C_k^{-1} A_k C_k, \dots \quad (14-12-1)$$

则所得矩阵序列 $\{A_k\}$ 收敛为分块三角形矩阵, 并且对角块都是 $1 \times 1$ 或 $2 \times 2$ 阶矩阵, 那么就可求出原始矩阵 $A$ 的全部特征值。

QR算法是每次迭代首先把 $A_k$ 分解成正交矩阵 $Q_k$ 和上三角矩阵 $R_k$ 的乘积, 即QR分解:

$$A_k = Q_k R_k \quad (14-12-2)$$

然后取变换矩阵 $C_k = Q_k$

从而由 $Q_k^{-1} = Q_k^T$ 和上式有:

$$A_{k+1} = Q_k^T A_k Q_k = R_k Q_k \quad (14-12-3)$$

从 $A_k$ 到 $A_{k+1}$ 的这种正交相似变换叫做QR变换。由于正交矩阵的行和列都是单位向量, 所以QR算法的显著优点是数值计算稳定, 但是就收敛性和每次迭代的计算量来说并不怎么好, 为此就用移动原点和两次QR法以解决这一矛盾。

从以上可知, QR算法的关键是 $A$ 矩阵的QR分解。QR分解是无条件的, 对此有定理1:

对于任何矩阵 $A$ (可以是复数矩阵), 存在正交矩阵 $Q$ 和对角线元素为非负实数的上三角矩阵 $R$ , 使分解式 $A=QR$ 成立。如果 $A$ 非奇异, 则分解是唯一的。

若 $A_1$ 是准三角阵, 则在一般情况下

[illegible]

$$\text{且 } |\lambda_1| > |\lambda_2| > \dots > |\lambda_n| > 0$$

这就是说,次对角元素  $a_{i, i-1}^{(k)}$  以因子  $\left| \frac{\lambda_i}{\lambda_{i-1}} \right|$  线性地收敛为零。对角线元素  $a_{i, i}^{(k)}$  以因子  $\max \left\{ \left| \frac{\lambda_i}{\lambda_{i-1}} \right|, \left| \frac{\lambda_{i-1}}{\lambda_i} \right| \right\}$  线性地收敛为特征值  $\lambda_i$ , 其中最后一对角线元素  $a_{nn}^{(k)}$  的收敛因子是  $\left| \frac{\lambda_n}{\lambda_{n-1}} \right|$ , 并且特征值按模大小在对角线上顺序排列。

在实际的计算中,线性收敛是不满意的。但是从上面的分析可以看出,如果每次迭代前将  $\mathbf{A}_k$  的特征值都减去一个尽量接近于  $\lambda_0$  的数值  $\mu^{(k)}$  (这相当于将特征值的坐标原点位移  $\mu^{(k)}$ ),并在迭代后加回去,那么收敛因子将减少为  $\left| \frac{\lambda_i - \mu^{(k)}}{\lambda_{i-1} - \mu^{(k)}} \right|$ 。特别是最后一个收敛

因子将变得更小:  $\left| \frac{\lambda_n - \mu^{(k)}}{\lambda_{n-1} - \mu^{(k)}} \right| \ll \left| \frac{\lambda_n}{\lambda_{n-1}} \right|$ , 从而使  $a_{n, n-1}^{(k)}$  迅速趋向零,  $a_{n, n}^{(k)}$  迅速趋向  $\lambda_n$ 。这就是用特征值原点位移加速算法收敛的基本思想。在上述特定的情况下, 作为  $\lambda_n$  的估计数值, 显然应取位移值  $\mu^{(k)} = a_{n, n}^{(k)}$ 。

所以，带原点位移的QR算法每次迭代分两步：

第一步, 对位移后的矩阵  $A_k - \mu^{(k)}I$  按式 (14-12-2) 和 (14-12-3) 作QR分解:

$$A_k - \mu^{(k)} I = Q_k R_k \quad (14-12-4)$$

然后作正交相似变换:

$$Q_k^*(A_k - \mu^{(k)}I)Q_k = R_k Q_k \quad (14-12-5)$$

第二步, 恢复特征值的原点, 即取矩阵

$$A_{k+1} = R_k Q_k + \mu^{(k)} I \quad (14-12-6)$$

显然  $A_k$  与  $A_{k+1}$  的特征值相同。

从式 (14-12-5) 可知,  $R_k Q_k = Q^* A_k Q_k - \mu^{(k)} I$  代入式 (14-12-6) 得出:

$$A_{k+1} = Q_k^* A_k Q_k \quad (14-12-7)$$

这就是说, 带原点位移的 QR 变换, 可先按 (14-12-4) 式进行 QR 分解, 矩阵  $A_k - \mu^{(k)} I$ , 然后用所得  $Q_k$  对  $A_k$  作正交相似变换, 即按 (14-12-7) 式求得  $A_{k+1}$ 。

也可以证明, 对于具有等模特征值的矩阵  $A = A_1$ , 由 QR 变换得到的矩阵序列  $\{A_k\}$  常常收敛于分块三角形, 并且每个对角块对应于模相等的特征值。特别地, 对于具有多重复共轭特征值的实矩阵,  $\{A_k\}$  的极限形式是分块三角形, 并且对角块为  $1 \times 1$  或  $2 \times 2$  阶矩阵, 其中  $2 \times 2$  阶对角块对应于一对复共轭特征值。

由此可见, 对于一般的矩阵, 特别是对于具有复数特征值的实矩阵, 采用加速最后一个对角线元素收敛的实数位移  $\mu^{(k)} = a_{nn}^{(k)}$  是不适宜的。事实上, 这时的  $\mu^{(k)}$  不可能接近于任何复数特征值。参虑到  $\{A_k\}$  的极限形式中最后一个对角块的阶数常常是  $2 \times 2$ , 作为改善办法, 可取  $A_k$  右下角子矩阵

$$\begin{pmatrix} a_{n-1, n-1}^{(k)} & a_{n-1, n}^{(k)} \\ a_{n, n-1}^{(k)} & a_{n, n}^{(k)} \end{pmatrix}$$

的两个特征值  $\mu_1^{(k)}$  和  $\mu_2^{(k)}$  中模值靠近  $a_{nn}^{(k)}$  的一个当作位移值  $\mu^{(k)}$ 。实践表明, 这种原点位移方法对加速 QR 算法收敛是很有效的。

对于实矩阵  $A = A_1$ , 当位移值为右下角  $2 \times 2$  阶子块的特征值时,  $\mu^{(k)}$  可能是复数, 从而按式 (14-12-4) 和式 (14-12-7) 进行的 QR 变换需作复数运算, 为了减轻计算量, 避免实矩阵的复数运算, 可采用把连续两次迭代合并为一次的二重 QR 算法。它的一般原理如下:

设经  $k-1$  次二重迭代后, 原始矩阵  $A_1$  已相似变换为  $A_{2k-1}$ 。对于第  $K$  次二重迭代, 首先从  $A_{2k-1}$  右下角  $2 \times 2$  阶子矩阵求出位移值  $\mu_1^{(k)}$  和  $\mu_2^{(k)}$ , 然后用它们按式 (14-12-4) 和式 (14-12-7) 连续作两次 QR 变换:

第一次, 对矩阵  $A_{2k-1}$  用位移值  $\mu_1^{(k)}$  作 QR 变换:

$$A_{2k-1} - \mu_1^{(k)} I = Q_{2k-1} R_{2k-1}$$

$$A_{2k} = R_{2k-1} Q_{2k-1} + \mu_1^{(k)} I = Q_{2k-1}^* A_{2k-1} Q_{2k-1}$$

第二次, 对矩阵  $A_{2k}$  用位移值  $\mu_2^{(k)}$  作 QR 变换:

$$A_{2k} - \mu_2^{(k)} I = Q_{2k} R_{2k}$$

$$A_{2k+1} = R_{2k} Q_{2k} + \mu_2^{(k)} I = Q_{2k}^* A_{2k} Q_{2k}$$

如果把这两次变换合并为一次, 并且令:

$$Q = Q_{2k-1} Q_{2k}, \quad R = R_{2k} R_{2k-1}$$

$$\varphi_2(A_{2k-1}) = (A_{2k-1} - \mu_1^{(k)} I)(A_{2k} - \mu_2^{(k)} I) \quad (14-12-8)$$

则由上列各式可推知:

$$\varphi_2(A_{2k-1}) = QR \quad (14-12-9)$$

$$A_{2k+1} = Q^* A_{2k-1} Q \quad (14-12-10)$$

由式 (14-12-8) 可见  $\varphi_2(A_{2k-1})$  是实矩阵, 于是由它分解出来的  $Q$  可取正交矩阵,

按式(14—12—10)得到的 $A_{n-k+1}$ 仍是实矩阵。因此对于实矩阵采用二重QR算法可避免因原点位移导致的复数运算。

综上所述,不可约准三角矩阵A的二重QR变换,最终可简化成两步:

第一步,用化 $\varphi_2(A)$ 第一列为上三角形的初等对称正交矩阵 $P_0$ ,对A作相似变换,得

$$B_1 = P_0 A P_0^T$$

第二步,用豪式方法化 $B_1$ 为准三角形,得二重变换的结果矩阵B,有关的计算公式如下,设矩阵A右下角 $2 \times 2$ 阶子矩阵

$$\begin{bmatrix} a_{n-1, n-1} & a_{n-1, n} \\ a_{n, n-1} & a_{n, n} \end{bmatrix}$$

的特征值为 $\mu_1$ 和 $\mu_2$ ,并令 $\delta = \mu_1 + \mu_2$ ,  $\rho = \mu_1 \cdot \mu_2$

$$\left. \begin{aligned} \delta &= a_{n-1, n-1} + a_{n, n} \\ \rho &= a_{n-1, n-1} a_{n, n} - a_{n-1, n} a_{n, n-1} \end{aligned} \right\} \quad (14-12-11)$$

因为A是准三角阵,故矩阵 $\varphi_2(A) = A^2 - \delta A + \rho I$ 的第一列只有前三个元素非零,设该列为

$$X_0 = (p_0, q_0, r_0, 0, \dots, 0)^T$$

$$\left. \begin{aligned} p_0 &= a_{11}(a_{11} - \delta) + a_{12}a_{21} + \rho \\ q_0 &= a_{21}(a_{11} + a_{22} - \delta) \\ r_0 &= a_{21}a_{22} \end{aligned} \right\} \quad (14-12-12)$$

矩阵 $P_0$ 应使 $X_0$ 经线性变换 $P_0 X_0$ 后消去其中的元素 $q_0$ 和 $r_0$ ,变换矩阵 $P_0$ 由下式给出:

$$P_0 = I - \frac{2}{\|V_0\|^2} V_0 V_0^T \quad (14-12-13)$$

$$\text{式中 } V_0 = ((p_0 + \sigma_0), q_0, r_0, 0, \dots, 0)^T$$

$$\text{而 } \sigma_0 = \text{sign}(p_0) \sqrt{p_0^2 + q_0^2 + r_0^2}$$

值得注意的是,这里 $P_0$ 具有分块对角形式,即:

$$P_0 = \begin{bmatrix} P_0^{(3)} & 0 \\ 0 & I_{n-3} \end{bmatrix}$$

其中 $P_0^{(3)}$ 是 $3 \times 3$ 阶矩阵。因此用 $P_0$ 对准三角形阵A作相似变换的结果为:

$$B_1 = P_0 A P_0^T = \begin{bmatrix} * & * & * & \cdots & * \\ p_1 & * & * & \cdots & * \\ \underline{q_1} & * & * & \cdots & * \\ \underline{r_1} & * & * & \cdots & * \\ & \ddots & \ddots & \ddots & \vdots \\ & & * & * & * \end{bmatrix}$$

其中打黑粗线的为次对角以下新增加的三个非零元素,用 $n-2$ 次豪式变换,将 $B_1$ 阵化为准三角阵B,即

$$B_2 = P_1 B_1 P_1^T, B_3 = P_2 B_2 P_2^T, \dots, B = B_{n-1} = P_{n-2} B_{n-2} P_{n-2}^T \quad (14-12-14)$$

这样依次化前 $n-2$ 列为准三角列。事实上,若令 $B_1$ 的第一列为 $X_1$ ,则矩阵 $P_1$ 应使线性变换 $P_1 X_1$ 消去 $X_1$ 的元素 $q_1$ 和 $r_1$ ,可见 $P_1$ 的算式以及分块对角形式与 $P_0$ 类同。按(14—12—14)式变换下去,一般地, $B_i$ 与 $B_1$ 一样,次对角线以下仅有三个非零元素(第 $i$ 列两



个, 第 $i+1$ 列一个), 即:

$$B_i = \begin{bmatrix} * & \dots & \dots & * \\ * & \ddots & & \vdots \\ & \ddots & * & * \\ & & \underline{p_i} & * \\ & & \underline{q_i} & * \\ & & \underline{r_i} & * \\ & & & \ddots \\ & & & * & * \end{bmatrix}$$

若令它的第 $i$ 列为 $X_i$ , 则矩阵 $P_i$ 应使线性变换 $P_i X_i$ 消去 $X_i$ 中的元素 $q_i$ 和 $r_i$ , 因此有:

$$P_i = I - \frac{2}{\|V_i\|^2} V_i V_i^T \quad (14-12-15)$$

$$\text{式中 } V_i = (0, \dots, 0, \underbrace{(p_i + \sigma_i)}_{i\uparrow}, q_i, r_i, 0, \dots, 0)^T$$

$$\text{而 } \sigma_i = \text{sign}(p_i) \sqrt{p_i^2 + q_i^2 + r_i^2}$$

式(14-12-15)就是求一般 $P_i$  ( $i=0, 1, 2, \dots, n-2$ )的一般计算公式, 其中计算 $P_0$ 的参数 $p_0, q_0, r_0$ 由式(14-12-11)和式(14-12-12)给出。计算 $P_i$ 的其它参数 $p_i, q_i$ 和 $r_i$ , 从相应的 $B_i$ 的第 $i$ 列取得

$$p_i = b_i^{(1)}{}_{1, i}, \quad q_i = b_i^{(1)}{}_{2, i}, \quad r_i = b_i^{(1)}{}_{3, i}$$

作为实准三角阵二重QR算法小结, 下面列出它的基本计算步骤:

(1) 找出次对角线倒数第一个等于零(或可忽略)的元素位置, 使算法对最后一个对角块——不可约准三角阵进行。

(2) 如果这一对角块的阶数是 $1 \times 1$ 或 $2 \times 2$ , 则求出其特征值, 并收缩原矩阵, 然后重复第1步; 否则按下列两步对它作二重QR变换。

(3) 按式(14-12-11)和(14-12-12)求出初始参数 $p_0, q_0$ 和 $r_0$ 。

(4) 按式(14-12-14)作 $n-1$ 次豪氏变换, 其中, 矩阵 $P_i$ 的算式是(14-12-15), 变换结束后返回第一步。

以上过程直到矩阵的阶数收缩为零, 即求出全部特征值时为止。

## 二、程序说明

1. 使用数组A, B, R, I。
2. 使用变量 I, J, E, T, N, N<sub>1</sub>, L, X, Y, W, T<sub>1</sub>, S, M, Z, P, Q, K, N<sub>2</sub>。
3. 程序结束, 原来信息破坏, 求出的特征值的实部在R数组中, 虚部在I数组中, B数组记迭代次数。
4. 共迭代六十次, 若仍不能收敛, 则程序失败, 印出: "FAIL"。
5. 运行结果中的TIMES项反映原矩阵的结构特征, 只是作为参考, 一般使用者可以不用管它。

## 三、使用说明

1. 用BASIC解释程序引入本程序, 键盘命令RUN使本程序投入运行, 机器印出,

"INPUT DEGREE N OF MATRIX H

2. 由键盘回答矩阵阶数N, 机器又自动印出:

"INPUT ELEM. OF MATRIX H"

3. 按行由键盘输入上H的元素, 则机器自动印出,

INPUT E=

4. 由键盘回答容许误差值E (例如可取 $E=0.1$ ,  $0.01$ 等), 最后机器自动印出结果, 各特征值顺序印出, 同时给出相应的迭代次数。

如果迭代不收敛, 印出 "FAIL" 失败。

#### 四、试题

$$H = \begin{pmatrix} 3 & 2 & 1 & 2 & 1 & 4 & 1 & 2 \\ 2 & 1 & 3 & 1 & 2 & 2 & 1 & 4 \\ & 3 & 1 & 2 & 1 & 2 & 1 & 3 \\ & & 1 & 1 & 2 & 1 & 3 & 1 \\ & & & 10^{-7} & 3 & 1 & 4 & 1 \\ & & & & 10^{-6} & 2 & 1 & 4 \\ & & & & & 1 & 2 & 3 \\ & & & & & & 3 & 2 \end{pmatrix}$$

特征值:

5.618185550	2.999999794
-2.429512472	0.354249176
1.94495965	$-0.4999510756 \times 10^{-6}$
0.86636719	5.645751608

#### 五、局限:

程序的失败是少见的, 当H阵类似正交阵时, 计算有可能失败或误差比较大。原因是机器字长太短造成的。对于一般工程实际使用, 这样的矩阵是极其例外的。

#### 六、程序清单及运行结果

程序清单

```

7130 REM HOR PROCEDURE
7132 PRINT "INPUT DEGREE OF MATRIX N"
7134 INPUT N
7136 PRINT
7138 PRINT "INPUT ELEM. OF MATRIX H"
7140 DIM A(N+4, N+4), B(N+4), R(N), I(N)
7142 FOR I=1 TO N
7144   FOR J=1 TO N
7146     INPUT A(I,J)
7148   NEXT J
7150   PRINT
7152 NEXT I
7154 LET E=.001

```

```

7156 LET T=0
7158 LET T2=N
7160 IF N=0 THEN GOTO 7410
7162 LET T1=0
7164 LET N1=N-1
7166 FOR L=N TO 2 STEP -1
7168   IF ABS(A(L,L-1))=<E*(ABS(A(L-1,L-1))+ABS(A(L,L))) THEN
       GOTO 7174
7170 NEXT L
7172 LET L=1
7174 LET X=A(N,N)
7176 IF L=N THEN GOTO 7354
7178 LET Y=A(N1,N1)
7180 LET W=(N,N1)*A(N1,N)
7182 IF L=N1 THEN GOTO 7364
7184 IF T1=60 THEN GOTO 7406
7186 IF T1=10 THEN GOTO 7192
7188 IF T1=20 THEN GOTO 7192
7190 GOTO 7208
7192 LET T=T+X
7194 FOR I=1 TO N
7196   LET A(I,I)=A(I,I)-X
7198 NEXT I
7200 LET S=ABS(A(N,N1))+ABS(A(N1,N-2))
7202 LET X=.75*S
7204 LET Y=.75*S
7206 LET W=-.4375*S*S
7208 LET T1=T1+1
7210 FOR M=N-2 TO L STEP -1
7212   LET Z=A(M,M)
7214   LET R=X-Z
7216   LET S=Y-Z
7218   LET P=(R*S-W)/A(M+1,M)+A(M,M+1)
7220   LET Q=A(M+1,M+1)-Z-R-S
7222   LET R=A(M+2,M+1)
7224   LET S=ABS(P)+ABS(Q)+ABS(R)
7226   LET P=P/S
7228   LET Q=Q/S
7230   LET R=R/S
7232   IF M=L THEN GOTO 7240

```

```

7234 LET U=E * ABS( P ) * (ABS(A(M-1, M-1))+ABS( Z )+ABS(A(M+1,
      M+1)))
7236 IF ABS(A(M, M-1)) * (ABS( Q )+ABS( R ))=<U THEN GOTO 7240
7238 NEXT M
7240 FOR I=M+2 TO N
7242 LET A(I, I-2)=0
7244 NEXT I
7246 FOR I=M+3 TO N
7248 LET A(I, I-3)=0
7250 NEXT I
7252 FOR K=M TO N1
7254 IF K=N1 THEN GOTO 7258
7256 LET N2=1
7258 IF K=M THEN GOTO 7282
7260 LET P=A(K, K-1)
7262 LET Q=A(K+1, K-1)
7264 IF N2=1 THEN GOTO 7270
7266 LET R=0
7268 GOTO 7272
7270 LET R=A(K+2, K-1)
7272 LET X=ABS(P)+ABS(Q)+ABS(R)
7274 IF X=0 THEN GOTO 7350
7276 LET P=P/X
7278 LET Q=Q/X
7280 LET R=R/X
7282 LET S=SQR(P * P+Q * Q+R * R)
7284 IF P=>0 THEN GOTO 7288
7286 LET S=-S
7288 IF K=M THEN GOTO 7294
7290 LET A(K, K-1)=-S * X
7292 GOTO 7298
7294 IF L=M THEN GOTO 7298
7296 LET A(K, K-1)=-A(K, K-1)
7298 LET P=P+S
7300 LET X=P/S
7302 LET Y=Q/S
7304 LET Z=R/S
7306 LET Q=Q/P
7308 LET R=R/P
7310 FOR J=K TO N

```

```

7312     LET P=A(K,J)+Q*A(K+1,J)
7314     IF N2=0 THEN GOTO 7320
7316     LET P=P+R*A(K+2,J)
7318     LET A(K+2,J)=A(K+2,J)-P*Z
7320     LET A(K+1,J)=A(K+1,J)-P*Y
7322     LET A(K,J)=A(K,J)-P*X
7324     NEXT J
7326     IF K+3=>N THEN GOTO 7332
7328     LET J=K+3
7330     GOTO 7334
7332     LET J=N
7334     FOR I=L TO J
7336         LET P=X*A(I,K)+Y*A(I,K+1)
7338         IF N2=0 THEN GOTO 7344
7340         LET P=P+Z*A(I,K+2)
7342         LET A(I,K+2)=A(I,K+2)-P*R
7344         LET A(I,K+1)=A(I,K+1)-P*Q
7346         LET A(I,K)=A(I,K)-P
7348     NEXT I
7350     NEXT K
7352     GOTO 7166
7354     LET R(N)=X+T
7356     LET I(N)=0
7358     LET B(N)=T1
7360     LET N=N1
7362     GOTO 7160
7364     LET P=(Y-X)/2
7366     LET Q=P*P+W
7368     LET Y=SQR(ABS(Q))
7370     LET B(N)=-T1
7372     LET B(N1)=T1
7374     LET X=X+T
7376     IF Q=<0 THEN GOTO 7394
7378     IF P=>0 THEN GOTO 7382
7380     LET Y=-Y
7382     LET Y=P+Y
7384     LET R(N1)=X+Y
7386     LET R(N)=X-W/Y
7388     LET I(N1)=0
7390     LET I(N)=0

```

```

7392 GOTO 7402
7394 LET R(N1)=X+P
7396 LET R(N)=X+P
7398 LET I(N1)=Y
7400 LET I(N)=-Y
7402 LET N=N-2
7404 GOTO 7160
7406 PRINT "FAIL"
7408 GOTO 7416
7410 FOR I=1 TO T2
7412 PRINT R(I),I(I),B(I)
7414 NEXT I
7416 END

```

程序运行结果

RUN

INPUT DEGREE N OF MATRIX H ? 8

INPUT ELEM. OF MATRIX H

```

? 3? 2? 1? 2? 1? 4? 1? 2
? 2? 1? 3? 1? 2? 2? 1? 4
? 0? 3? 1? 2? 1? 2? 1? 3
? 0? 0? 1? 1? 2? 1? 3? 1
? 0? 0? 0? 1E-7? 3? 1? 4? 1
? 0? 0? 0? 0? 1E-6? 2? 1? 4
? 0? 0? 0? 0? 0? 1? 2? 3
? 0? 0? 0? 0? 0? 0? 3? 2

```

INPUT E=? .0001

EIGENVALUES OF THE MATRIX H ARE

REAL	IMAGINARY	TIMES
5.61818	0	0
-2.42951	0	0
1.94496	0	3
.866367	0	-3
3	0	0
.354252	0	0
3.93111E-5	0	0
5.64571	0	5

END AT 7416

## §13 用克雷洛夫方法求矩阵的特征多项式

### 一、计算方法概要

设矩阵  $A$  的特征多项式为

$$\Psi(\lambda) = (-1)^n (\lambda^n - p_1 \lambda^{n-1} - \dots - p_{n-1} \lambda - p_n) \quad (14-13-1)$$

由凯莱-哈密顿 (Cayley-Hamilton) 定理有:

$$\Psi(A) = (-1)^n (A^n - p_1 A^{n-1} - \dots - p_{n-1} A - p_n I) = 0 \quad (14-13-2)$$

将 (14-13-2) 式两边乘以任意非零向量  $C_0$ , 就得到:

$$A^n C_0 - p_1 A^{n-1} C_0 - \dots - p_{n-1} A C_0 - p_n C_0 = 0 \quad (14-13-3)$$

如果记  $A^k C_0 = C_k$ , 则 (14-13-3) 式可以改写为:

$$p_1 C_{n-1} + p_2 C_{n-2} + \dots + p_n C_0 = C_n \quad (14-13-4)$$

对任意给定的  $C_0$ , 可以用下面递推公式依次算出  $C_1, C_2, \dots, C_n$ :

$$C_k = A C_{k-1} \quad (k=1, 2, \dots, n) \quad (14-13-5)$$

把这些数值代入到 (14-13-4) 式中去, 就得到一组线性代数方程组, 可以用解线性方程组的方法求出特征多项式的系数  $p_1, p_2, \dots, p_n$ . 由特征多项式可以用解代数高次方程方法求出矩阵的特征值.

可能出现方程组 (14-13-4) 的系数行列式等于零的情况; 于是不能从 (14-13-4) 中得到唯一解  $p_1, p_2, \dots, p_n$ . 此时 (14-13-4) 式中系数行列式中各列  $C_0, C_1, \dots, C_{n-1}$  是线性组合, 则  $C_{k+1}, \dots, C_{n-1}$  也都是  $C_0, C_1, \dots, C_{k-1}$  的线性组合, 所以  $C_0, C_1, \dots, C_{n-1}$  中线性无关的向量一定是前面连续的几个, 因此我们可以假定有一个最大的正整数  $k$  存在, 使得  $C_0, C_1, \dots, C_{k-1}$  是线性无关的从而有关系式:

$$q_1 C_{k-1} + q_2 C_{k-2} + \dots + q_k C_0 = C_k \quad (14-13-6)$$

对于给定的  $C_0$  而言, 关系式 (14-13-6) 是唯一确定的. 如果记:

$$\Phi(\lambda) = \lambda^k - q_1 \lambda^{k-1} - q_2 \lambda^{k-2} - \dots - q_k$$

则  $\Phi(\lambda)$  是  $\Psi(\lambda)$  的因式, 实际上若

$$\Psi(\lambda) = \theta(\lambda) \Phi(\lambda) + \gamma(\lambda) \quad (\gamma(\lambda) \text{ 的次数小于 } k)$$

$$\Psi(A) C_0 = 0$$

$$\theta(A) \cdot \Phi(A) \cdot C_0 = 0$$

$$\therefore \gamma(A) \cdot C_0 = 0$$

$\Phi(\lambda)$  既是  $\Psi(\lambda)$  的因式, 因此  $\Phi(\lambda) = 0$  的根, 也就是特征值. 不过由  $\Phi(\lambda) = 0$  得到的特征值不一定是全部特征值.

在实际计算中, 我们事先并不知道  $C_0, C_1, \dots, C_{n-1}$  中线性无关向量的个数, 不过, 当我们用高斯消去法解方程组 (14-13-4) 时, 在计算过程中能确定出来.

### 二、程序说明

本程序使用的工作单元:

$B(N-1, N-1)$  为线性方程组的增广矩阵

$A(N-1, N)$  为中间运算数组

$N, I, I_1, I_2, I_3, N_1$  为循环变量或简单变量

### 三、上机操作步骤

1. 用 BASIC 解释程序将本程序输入机内
2. 用键盘命令 RUN 启动本程序，本程序开始运行，在电传上打印出：  
"INPUT MATRIX DIMENSION N"
3. 用户根据需要输入矩阵的阶数，回答完阶数后应按回车键。然后电传机又打印出：  
"INPUT MATRIX IN ROW"
4. 要求用户按行输入矩阵，每输入一个数要按一下回车键。直至全部输入。
5. 全部输入完毕后，计算机输出两项内容：第一项为求出的线性方程组的系数增广矩阵。开头以  
"THE AUGMENTED MATRIX A IS"  
按标准格式打印出来。  
第二项为特殊多项式：  
"THE PROPER POLYNOMIAL IS"  
在打印出来的特征多项式中，因为没有符号( $\lambda$ )这个字母，所以用英文字母"L"来代替" $\lambda$ "

#### 四、试题

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1E-3 & 3E-3 & 2E-3 & 1E-3 \\ 0 & 1E-6 & 3E-6 & 2E-6 \\ 0 & 0 & 1E-9 & 2E-9 \end{bmatrix}$$

计算结果：

系数增广矩阵

$$\begin{bmatrix} 1.00401 & 1.002 & 1 & 1 & -1.00602 \\ 1.00501E-3 & 0.001 & 0.001 & 0 & -1.00702E-3 \\ 1.003E-9 & 0 & 0 & 0 & -1.00501E-9 \\ 0 & 0 & 0 & 0 & -1.003E-18 \end{bmatrix}$$

特征多项式为：

$$L^4 - 1.003L^3 + (1.0004E-3)L^2 + (3.94881E-7)L - (1.39715E-6)$$

#### 五、程序清单及运行结果

程序清单

```

7420 REM PROGRAM FINDING THE PROPER POLYNOMIAL OF MATRIX
7422 LET M=0
7424 PRINT "INPUT MATRIX DIMENSION N";
7426 INPUT N
7428 PRINT
7430 PRINT
7432 DIM A(N-1,N),B(N-1,N-1)
7434 PRINT "INPUT MATRIX IN ROW"
7436 FOR I=0 TO N-1
7438   FOR I1=0 TO N-1
7440     INPUT B(I,I1)

```



```

7442     LET A(I,I1)=0
7444     NEXT I1
7446     PRINT
7448     LET A(I,N)=0
7450     NEXT I
7452     LET A(0,N-1)=1
7454     FOR I2=N-1 TO 0 STEP -1
7456         FOR I=0 TO N-1
7458             FOR I1=0 TO N-1
7460                 IF I2=0 THEN GOTO 7466
7462                 LET A(I,I2-1)=A(I,I2-1)+B(I,I1)*A(I1,I2)
7464                 GOTO 7468
7466             LET A(I,N)=A(I,N)-B(I,I1)*A(I1,0)
7468             NEXT I1
7470         NEXT I
7472     NEXT I2
7474     PRINT
7476     PRINT "THE AUGMENTED MATRIX A IS"
7478     FOR I=0 TO N-1
7480         FOR I1=0 TO N
7482             PRINT TAB(12*I1); A(I,I1);
7484         NEXT I1
7486     PRINT
7488     NEXT I
7490     IF N=<4 THEN GOTO 7496
7492     PRINT "MATRIX A TOO BIG TO PRINT IT IN",
7494     PRINT "STANDARD FORMAT,EXECUSED"
7496     PRINT
7498     FOR I=0 TO N-1
7500         IF I=N-1 THEN GOTO 7528
7502         LET I3=A(I,I)
7504         LET I2=I
7506         FOR I1=I+1 TO N-1
7508             IF ABS(I3)>ABS(A(I1,I)) THEN GOTO 7514
7510             LET I3=A(I1,I)
7512             LET I2=I1
7514         NEXT I1
7516         IF ABS(I3)>1E-30 THEN GOTO 7526
7518         PRINT "THE PROPER POLYNOMIAL HAVE", N-I,"MULTIROOTS"
7520         PRINT "THE FOLLOWING POLYNOMIAL IS NOT PROPER ONE"

```

```

7522     GOTO 7552
7524     IF I2=I THEN GOTO 7528
7526     GOSUB 7600
7528     FOR I1=I+1 TO N
7530         LET A(I,I1)=A(I,I1)/A(I,I)
7532     NEXT I1
7534     IF I=N-1 THEN GOTO 7548
7536     FOR I1=I+1 TO N-1
7538         FOR I2=I+1 TO N
7540             LET A(I1,I2)=A(I1,I2)-A(I1,I)*A(I,I2)
7542         NEXT I2
7544     NEXT I1
7546 NEXT I
7548 LET I=I+1
7550 PRINT "THE PROPER POLYNOMIAL IS"
7552 LET N1=N
7554 LET N=I
7556 FOR I2=N-2 TO 0 STEP -1
7558     FOR I1=I2+1 TO N-1
7560         LET A(I2,N1)=A(I2,N1)-A(I2,I1)*A(I1,N1)
7562     NEXT I1
7564 NEXT I2
7566 PRINT
7568 LET I3=1
7570 IF N1=2*INT(N1/2) THEN GOTO 7576
7572 PRINT "-";
7574 LET I3=-1
7576 PRINT "L(";N;")";
7578 FOR I=0 TO N-2
7580     IF I3*A(I,N1)<0 THEN GOTO 7584
7582     PRINT "+";
7584     PRINT I3*A(I,N1);"*L(";N-I-1;")";
7586 NEXT I
7588 IF I3*A(N-1,N1)<0 THEN GOTO 7592
7590 PRINT "+";
7592 PRINT I3*A(N-1,N1)
7594 PRINT
7596 IF M=0 THEN GOTO 7612
7598 RETURN
7600     FOR I1=I TO N

```

```

7602 LET I3=A(I2,I1)
7604 LET A(I2,I1)=A(I,I1)
7606 LET A(I,I1)=I3
7608 NEXT I1
7610 RETURN
7612 END

```

程序运行结果

RUN

INPUT MATRIX DIMENSION N ? 4

INPUT MATRIX IN ROW

```

? 1 ? 2 ? 3 ? 4
? 1E-3 ? 3E-3 ? 2E-3 ? 1E-3
? 0 ? 1E-6 ? 3E-6 ? 2E-6
? 0 ? 0 ? 1E-9 ? 2E-9

```

THE AUGMENTED MATR IX A IS

1.00401	1.002	1	1	-1.00602
1.00501E-3	.001003	.001	0	-1.00702E-3
1.003E-9	1E-9	0	0	-1.00501E-9
1E-18	0	0	0	-1.003E-18

THE PROPER POLYNOMIAL IS

$L(4) - 1.003 L(3) + 1.0004 E-3 L(2) + 3.94881 E-7 L(1) - 1.39715 E-6$

END AT 7612

### 本章参考文献

[1] 南京大学数学系计算数学专业编

《线性代数计算方法》

科学出版社 1979.

[2] 《电子计算机常用算法》

科学出版社 1976.

[3] 西安交通大学编

《电力系统计算》

水利电力出版社 1978.

## 第十五章 多元线性回归分析

### 一、基本概念和计算方法

设随机变量 $Y$ 随自变量 $X_1, X_2, \dots, X_n$ 变化, 有 $m$ 组观测数据 $(X_{1t}, X_{2t}, \dots, X_{nt}, Y_t), t=1, 2, \dots, m$ . 若要确定随机变量 $Y$ 与各个变量 $X_1, X_2, \dots, X_n$ 是否存在相关关系, 若存在的话给出合适的线性组合关系式. 这就是线性回归分析主要解决的问题之一. 那么怎样选择合适的表达式呢? 一般采用最小二乘法, 具体计算方法如下.

若用如下线性组合多项式表示随机变量 $Y$ 与自变量 $X_1, X_2, \dots, X_n$ 之间的关系:

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \dots + b_n X_n \quad (15-1)$$

将以上 $m$ 组 $X_1, X_2, \dots, X_n$ 观测数据代入(15-1)式, 可以得到 $m$ 个 $Y$ 的计算值记为 $\hat{Y}$ , 用这 $m$ 个计算值与 $Y$ 的 $m$ 个观测值求差的平方和, 并使之最小, 来确定 $b_0, b_1, b_2, \dots, b_n$ 各个参数. 这就叫做最小二乘法, 而 $b_0, b_1, \dots, b_n$ 叫做回归系数. 用数学式子表示如下:

$$Q = \sum_{t=1}^m (Y_t - \hat{Y}_t)^2 = \sum_{t=1}^m [Y_t - (b_0 + b_1 X_{1t} + \dots + b_n X_{nt})]^2 = \min \quad (15-2)$$

由数学分析的极值原理知, 要使 $b_i$ 满足以下方程组:

$$\frac{\partial Q}{\partial b_i} = 0, \quad j=0, 1, \dots, n \quad (15-3)$$

则 $b_i$ 就是使 $Q$ 最小的所求各个回归系数.

由 $\frac{\partial Q}{\partial b_0} = 0$  可以求得

$$\sum_{t=1}^m [Y_t - (b_0 + b_1 X_{1t} + \dots + b_n X_{nt})] = 0$$

$$\text{即 } b_0 = \bar{Y} - (b_1 \bar{X}_1 + b_2 \bar{X}_2 + \dots + b_n \bar{X}_n)$$

$$b_0 = \bar{Y} - \sum_{i=1}^n b_i \bar{X}_i \quad (15-4)$$

$$\text{其中 } \bar{Y} = \frac{1}{m} \sum_{t=1}^m Y_t$$

$$\bar{X}_j = \frac{1}{m} \sum_{t=1}^m X_{jt}, \quad j=1, 2, \dots, n$$

将(15-4)代入(15-2)可以得

$$Q = \sum_{t=1}^m [Y_t - \bar{Y} - \sum_{i=1}^n b_i (X_{it} - \bar{X}_i)]^2$$

将上式代入(15-3), 且取 $j=1, 2, \dots, n$ , 就可得到 $n$ 个联立方程组, 即:

$$\sum_{i=1}^m 2 \left\{ (Y_i - \bar{Y} - \sum_{j=1}^n b_j (X_{ij} - \bar{X}_j)) (-\sum_{j=1}^n (X_{ij} - \bar{X}_j)) \right\} = 0$$

整理后得

$$\sum_{i=1}^n \left[ \sum_{j=1}^m (X_{ij} - \bar{X}_j)(X_{it} - \bar{X}_i) \right] b_j = \sum_{i=1}^m (Y_i - \bar{Y})(X_{it} - \bar{X}_i) \quad (15-5)$$

其中  $i=1, 2, \dots, n$

把 (15-5) 式写得更直观一点如下:

$$\begin{cases} l_{11}b_1 + l_{12}b_2 + \dots + l_{1n}b_n = l_{1y} \\ l_{21}b_1 + l_{22}b_2 + \dots + l_{2n}b_n = l_{2y} \\ \dots \dots \dots \\ l_{n1}b_1 + l_{n2}b_2 + \dots + l_{nn}b_n = l_{ny} \end{cases}$$

其中

$$\begin{aligned} l_{ij} &= \sum_{t=1}^m (X_{it} - \bar{X}_i)(X_{it} - \bar{X}_j) \\ &= \sum_{t=1}^m X_{it} \cdot X_{jt} - \frac{1}{m} \left[ \sum_{t=1}^m X_{it} \right] \cdot \left[ \sum_{t=1}^m X_{jt} \right] \end{aligned} \quad (15-6)$$

$$\begin{aligned} l_{iy} &= \sum_{t=1}^m (Y_t - \bar{Y})(X_{it} - \bar{X}_i) \\ &= \sum_{t=1}^m Y_t \cdot X_{it} - \frac{1}{m} \left[ \sum_{t=1}^m X_{it} \right] \cdot \left[ \sum_{t=1}^m Y_t \right] \end{aligned} \quad (15-7)$$

(15-5) 式给出的联立方程组称为“正规方程”或“法方程”。通过高斯消元法解此线性方程组就可将诸  $b_i$  求出, 而  $b_0$  可以通过 (15-4) 式求出。以上就是本程序的基本计算方法。

通过以上方法求出的各回归系数是不是可信呢? 为此还要进行方差分析。所谓方差分析就是要统计几个统计量, 它们是  $Y$  的残差平方和  $Q$ , 回归平方和  $U$ 、剩余标准差  $S$  和相关系数  $R$ 。它们分别定义如下:

$$\text{残差平方和 } Q = \sum_{i=1}^m (Y_i - \hat{Y}_i)^2$$

$$\text{回归平方和 } U = \sum_{i=1}^m (\hat{Y}_i - \bar{Y})^2$$

而残差平方和  $Q$  与回归平方和  $U$  相加之和, 叫做  $Y$  的离差平方和, 用  $S_{yy}$  表示。

$$\text{即 } S_{yy} = Q + U \quad (15-8)$$

$$\text{而 } S_{yy} = \sum_{i=1}^m (Y_i - \bar{Y})^2 = \sum_{i=1}^m Y_i^2 - \frac{1}{m} \left[ \sum_{i=1}^m Y_i \right]^2 \quad (15-9)$$

$$\begin{aligned} \text{而 } U &= \sum_{i=1}^m (\hat{Y}_i - \bar{Y})(Y_i - \bar{Y}) \\ &= \sum_{i=1}^m \sum_{j=1}^n b_j (X_{ij} - \bar{X}_j)(Y_i - \bar{Y}) \end{aligned}$$

$$= \sum_{i=1}^n b_i l_{iy} \quad (15-10)$$

Q也可按下列简便式子计算：

$$Q = S_{yy} - U = S_{yy} - \sum_{i=1}^n b_i l_{iy} \quad (15-11)$$

用(15—9)、(15—10)和(15—11)式分别计算 $S_{yy}$ ,  $U$ ,  $Q$ 可使计算量大大减小。 $U$ 值的大小反映了 $Y$ 总的变差中由于 $X_i$ 和 $Y$ 线性关系而引起的可变化部分,  $Y$ 的这部分变差是可以通过控制 $X_i$ 的取值而避免的, 是可控程度的表示。而 $Q$ 值大小反映了所有观测点距回归曲线的残差的平方和, 根据最小二乘原则, 这个量是在所有类似的曲线中与观测点距离的平方和最小的一个。它是除了 $X_i$ 对 $Y$ 的线性影响之外的一切因素(包括 $X_i$ 对 $Y$ 的非线性影响及测量误差等)对 $Y$ 变差的作用。 $Y$ 的这部分变差不能通过选择 $X_i$ 取值而加以改变, 因此是不可控部分。

结论:  $U$ 愈大(或 $Q$ 愈小)则表示 $Y$ 与这些自变量的线性关系愈密切, 回归的规律性愈强, 回归出来的结果可信程度愈高。

剩余平方和 $Q$ 除以它的自由度( $m-n+1$ )所得到的商叫剩余均方(或剩余方差)剩余方差的平方根称为剩余标准差, 用 $S$ 表示, 即:

$$S = \sqrt{\frac{Q}{m-n-1}} \quad (15-12)$$

$S$ 值大小可以看作是排除了 $X_i$ 对 $Y$ 的线性影响之后(或者说当 $X_i$ 取值固定时)衡量 $Y$ 值随机波动的大小的一个估计值。

根据正态分布的性质, 对于固定的 $X_i$ 值,  $Y$ 的取值是以 $\hat{Y}_0$ 为中心而对称分布的。愈靠近 $\hat{Y}_0$ 的地方出现的机会愈大, 而离 $\hat{Y}_0$ 较远的地方出现的机会就愈小, 且与剩余标准差 $S$ 之间的关系如下:

落在 $\hat{Y}_0 \pm 0.5S$ 的概率为0.38 ;

落在 $\hat{Y}_0 \pm S$ 的概率为0.68 ;

落在 $\hat{Y}_0 \pm 2S$ 的概率为0.95 ;

落在 $\hat{Y}_0 \pm 3S$ 的概率为0.997 。

由此可见 $S$ 愈小, 则从回归方程预报 $Y$ 就愈精确。因此剩余标准差作为预报精度的标志。在实际的使用中只要比较 $S$ 值与允许的偏差就行。所以 $S$ 值是检验一个回归是否有效的极其重要的标志。

相关系数 $R$ 定义为:

$$R = \sqrt{\frac{U}{S_{yy}}} = \sqrt{\frac{S_{yy} - Q}{S_{yy}}} = \sqrt{1 - \frac{Q}{S_{yy}}} \quad (15-13)$$

由(15—13)式知,  $R^2$ 实质上是 $Y$ 的回归平方和 $U$ 与 $Y$ 的离差平方和 $S_{yy}$ 的比值, 这个比值大小表示了 $Y$ 与 $X_1, X_2, \dots, X_n$ 的线性关系的密切程度。显然 $0 \leq R \leq 1$ ,  $R$ 越接近于1, 表明 $Y$ 与 $X_1, X_2, \dots, X_n$ 的线性关系越强。一般我们对 $R$ 不进行显著性检验, 而是利用 $F$ 值对整个回归进行显著性检验。 $F$ 值如下定义:

$$F = \frac{U/n}{Q/(m-n+1)} \quad (15-14)$$

通过查F分布表进行显著性检验，可以较准确地判断Y与所考虑的n个变量 $X_1, X_2, \dots, X_n$ 之间的线性关系究竟是否显著。

## 二、程序说明

本程序由两部分组成，其中第一部分由8002到8062语句组成，是高斯消元法解线性方程组子程序；第二部分是从8064到8224语句组成，是最小二乘和方差分析工作程序。

本程序使用数组F、A、B和W；使用简单变量有C、T、J、B、 $U_1$ 、 $U_2$ 、 $U_3$ 、 $U_4$ 、 $U_5$ 、L、S、R、F、Q、U。

程序结束，原始信息没有破坏，存放在F阵中：

$$F = \begin{pmatrix} * & \dots & * \\ \vdots & & \vdots \\ * & \dots & * \end{pmatrix} \quad \begin{array}{l} \text{—— Y 的观测数据。} \\ \vdots \\ \text{—— } X_i \text{ 的观测数据。} \end{array}$$

## 三、程序使用上机操作

1. 由BASIC引入本程序，用键盘命令RUN启动本程序，机器自动印出：

“INPUT NUM. OF VARIABLE N”

询问自变量的个数。

2. 用键盘回答自变量的个数后，机器又自动印出：

“INPUT NUM. OF DATA M”

询问数据组数。

3. 用键盘回答数据组数后，机器又自动印出：

“INPUT DATA Y,  $X_1, X_2, \dots, X_N$ ”

要求使用者从键盘输入数据。数据输入的次序是先把所有Y的观测数据逐个输入完后，再依次输入 $X_1, X_2$ 等的所有观测数据。

4. 数据输入完，机器进行计算，计算完后就将Y与各自变量的线性组合关系式印出：

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \dots + b_n X_n$$

然后分别印出方差分析的几个统计参数U、Q、F、R、S各值。

## 四、试题：

见中国科学院数学所编 科学出版社 1974 出版

《回归分析方法》P63页例7.2

结果应为： $Y = 0.6974 + 0.1606X_1 + 0.1076X_2 + 0.0359X_3$

## 五、程序清单及运行结果

程序清单

```
8000 GOTO 8064
8002 PRINT
8004 FOR S=1 TO N
8006   FOR T=S TO N
8008     IF A(T,S)<0 THEN GOTO 8016
8010   NEXT T
8012 PRINT "NO UNIQUE SOLUTION"
```

```

8014   GOTO 8062
8016   GOSUB 8036
8018   LET C=1/A(S,S)
8020   GOSUB 8048
8022   FOR T=1 TO N
8024       IF T=S THEN GOTO 8030
8026       LET C=-A(T,S)
8028       GOSUB 8056
8030   NEXT T
8032   NEXT S
8034   GOTO 8062
8036   FOR J= 1 TO N+ 1
8038       LET B=A(S,J)
8040       LET A(S,J)=A(T,J)
8042       LET A(T,J)=B
8044   NEXT J
8046   RETURN
8048   FOR J= 1 TO N+ 1
8050       LET A(S,J)=C * A(S,J)
8052   NEXT J
8054   RETURN
8056   FOR J= 1 TO N+ 1
8058       LET A(T,J)=A(T,J)+C * A(S,J)
8060   NEXT J
8062   RETURN
8064   PRINT "INPUT NUM. OF VARIABLE N"
8066   INPUT N
8068   PRINT
8070   PRINT "INPUT NUM. OF DATA M"
8072   INPUT M
8074   PRINT
8076   DIM F(N,M),A(N+ 1,N+ 1),B(N+ 1,N+ 1)
8078   PRINT " INPUT DATA Y,X1,X2,...,XN"
8080   PRINT "Y"
8082   FOR I= 0 TO N
8084       IF I= 0 THEN GOTO 8088
8086       PRINT "X", I
8088       FOR J= 0 TO M- 1
8090           INPUT F(I,J)
8092       NEXT J

```



```

8094 PRINT
8096 NEXT I
8098 FOR I= 0 TO N- 1
8100 LET U4= 0
8102 LET U5= 0
8104 FOR J= 0 TO N- 1
8106 LET U1= 0
8108 LET U2= 0
8110 LET U3= 0
8112 FOR K= 0 TO M- 1
8114 LET U1=U1+F(I+1,K)
8116 LET U2=U2+F(J+1,K)
8118 LET U3=U3+F(I+1,K) * F(J+1,K)
8120 NEXT K
8122 IF J<I THEN GOTO 8128
8124 LET A(I+ 1 ,J+ 1 )=U3-(U1 * U2)/M
8126 LET A(J+ 1 ,I+ 1 )=A(I+ 1 ,J+ 1 )
8128 NEXT J
8130 FOR K= 0 TO M- 1
8132 LET U4=U4+F(I+ 1 ,K) * F( 0 ,K)
8134 LET U5=U5+F( 0 ,K)
8136 NEXT K
8138 LET A(I+ 1 ,N+ 1 )=U4-(U1 * U5)/M
8140 NEXT I
8142 FOR I= 1 TO N
8144 FOR J= 1 TO N+ 1
8146 LET B(I,J)=A(I,J)
8148 NEXT J
8150 NEXT I
8152 GOSUB 8002
8154 FOR I= 0 TO N
8156 LET W(I)= 0
8158 FOR K= 0 TO M- 1
8160 LET W(I)=W(I)+F(I,K)
8162 NEXT K
8164 LET W(I)=W(I)/M
8166 NEXT I
8168 LET L= 0
8170 FOR I= 1 TO N
8172 LET L=L+W(I) * A( I ,N+ 1 )

```

```

8174 NEXT I
8176 LET B0=W(0)-L
8178 PRINT "Y=";B0;
8180 FOR I=1 TO N
8182   PRINT "+",A(I,N+1),"X",I;
8184 NEXT I
8186 PRINT
8188 LET U=0
8190 FOR I=1 TO N
8192   LET U=U+A(I,N+1)*B(I,N+1)
8194 NEXT I
8196 LET U1=0
8198 LET U2=0
8200 FOR K=0 TO M-1
8202   LET U1=U1+F(0,K)*F(0,K)
8204 NEXT K
8206 LET Q=U1-(U5*U5)/M-U
8208 LET S=SQR(Q/(M-N-1))
8210 LET R=SQR(1-Q/(U1-(U5*U5)/M))
8212 LET F1=(M-N-1)*U/(N*Q)
8214 PRINT "U=";U
8216 PRINT "Q=";Q
8218 PRINT "F=";F1
8220 PRINT "S=";S
8222 PRINT "R=";R
8224 END

```

运行结果

RUN

INPUT NUM. OF VARIABLE N

? 3

INPUT NUM.OF DATA M

? 49

INPUT DATA Y,X1,X2...,XN

Y

? 4.33 ? 3.65 ? 4.48 ? 5.55 ? 5.50 ? 3.11 ? 5.11 ? 3.88 ? 4.67 ? 4.95  
 ? 5.00 ? 5.27 ? 5.37 ? 5.48 ? 4.60 ? 5.66 ? 6.08 ? 3.22 ? 5.81 ? 4.73  
 ? 4.68 ? 3.12 ? 2.61 ? 3.72 ? 3.89 ? 2.70 ? 5.63 ? 5.81 ? 5.13 ? 5.39  
 ? 4.46 ? 4.66 ? 4.52 ? 4.87 ? 5.36 ? 4.61 ? 2.38 ? 3.87 ? 4.59 ? 5.16  
 ? 5.44 ? 4.00 ? 4.40 ? 4.06 ? 2.29 ? 4.71 ? 4.53 ? 5.36 ? 6.08

X 1

? 2 ? 7 ? 5 ? 12 ? 1 ? 3 ? 3 ? 6 ? 7 ? 0 ? 3 ? 0 ? 8 ? 6 ? 0 ? 3 ? 7  
? 16 ? 6 ? 0 ? 9 ? 4 ? 0 ? 9 ? 2 ? 9 ? 12 ? 6 ? 12 ? 0 ? 5 ? 4 ? 0 ? 6  
? 4 ? 10 ? 4 ? 5 ? 9 ? 6 ? 5 ? 5 ? 8 ? 2 ? 7 ? 4 ? 10 ? 3 ? 4

X 2

? 18 ? 9 ? 14 ? 3 ? 20 ? 12 ? 17 ? 5 ? 8 ? 23 ? 16 ? 18 ? 4 ? 14  
? 21 ? 14 ? 12 ? 0 ? 16 ? 15 ? 0 ? 6 ? 17 ? 0 ? 16 ? 6 ? 5 ? 13 ? 7  
? 24 ? 12 ? 15 ? 20 ? 16 ? 17 ? 4 ? 14 ? 13 ? 8 ? 13 ? 8 ? 11 ? 6 ? 13  
? 8 ? 10 ? 5 ? 17 ? 15

X 3

? 50 ? 40 ? 46 ? 43 ? 64 ? 40 ? 64 ? 39 ? 37 ? 55 ? 60 ? 49 ? 50  
? 51 ? 51 ? 51 ? 56 ? 48 ? 45 ? 52 ? 40 ? 32 ? 47 ? 44 ? 39 ? 39 ? 51  
? 41 ? 47 ? 61 ? 37 ? 49 ? 45 ? 42 ? 48 ? 48 ? 36 ? 36 ? 51 ? 54  
? 100 ? 44 ? 63 ? 55 ? 50 ? 45 ? 40 ? 64 ? 72

$Y = .69094 + .160829X_1 + .107657X_2 + 3.59783E-2X_3$

$U = 15.2567$

$Q = 29.6781$

$F = 7.7111$

$S = .812104$

$R = .582692$

END AT 8224

### 参考文献

- [1] 中国科学院教学所编《回归分析方法》1974.
- [2] 中国科学院教学所编《常用数理统计方法》1973.

## 第十六章 一维最优化方法

求多维函数 $f(X)$  ( $X$ 为 $N$ 维向量)的极值点问题,常按以下格式进行:选定初始点 $X_0$ ,从 $X_0$ 出发沿某一规定方向 $P_0$ 求函数 $f$ 的极值点,设其为 $X_1$ ;然后再从 $X_1$ 出发,沿某一规定方向 $P_1$ 求函数 $f$ 的极值点,设其为 $X_2$ 。如此继续,一般说来,从点 $X_k$ 出发,沿规定方向 $P_k$ 求函数 $f$ 的极值点 $X_{k+1}$ ,就构成某些最优化方法的一种基本过程。我们称 $P_k$ 为 $K$ 次迭代的寻查方向,它为具体的算法所规定。在过 $X_k$ 点的 $P_k$ 方向,函数 $f$ 实际上变成了参变量 $t$ 的一元函数 $f(X_k + tP_k)$ ,因为无论 $t$ 取什么值,点 $(X_k, P_k)$ 总位于过 $X_k$ 点的 $P_k$ 方向(见下图),这种求 $t$ 使 $f(X_k + tP_k)$ 等于极小的过程,叫做一维寻优或线性寻优。

根据前面的说明,不难理解,一维寻优的方法是实现某些最优化方法的一种基本方法,其重要性可想而知。在这里我们主要涉及三个方法:

- ①直接优选法
- ②二次插值法
- ③有理插值法

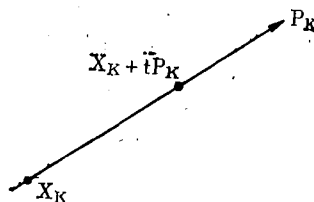


图 16-0-1

### §1 直接优选法求极值

#### 一、算法概要:

已知 $f(x)$ 在 $[a, b]$ 区间是单峰性函数,确有唯一极小点。则用0.618法逐步缩小 $(a, b)$ 区间,在给定精确度 $\varepsilon > 0$ 范围内,经过多次反复,总可以找出 $f(x)$ 在 $[a, b]$ 区间的近似的极小值。

计算步骤如下:

(1) 给出区间值 $A_1, A_2, (A_1 < A_2)$ 和精确度 $E (E > 0)$

(2) 计算  $A_3 = A_1 - 0.618(A_2 - A_1)$

$$A_4 = A_1 + 0.618(A_2 - A_1)$$

$$F_3 = f(A_3), F_4 = f(A_4)$$

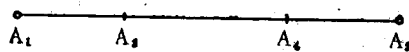


图 16-1-1

(3) 比较 $F_3$ 和 $F_4$ ,

(4) 若 $F_3 = F_4$ ,把区间缩小为 $A_3, A_4$ .

即令:  $A_1 = A_3, A_2 = A_4$ . 重复(2),继续进行比较。

若 $F_3 < F_4$ ,把区间缩小为 $A_1, A_4$ .

即令:  $A_2 = A_4, A_1 = A_1, F_4 = F_3$

$$A_3 = A_1 - 0.618(A_2 - A_1)$$

$$F_3 = f(A_3)$$

重复(3), 继续进行比较

若  $F_3 > F_4$ , 缩小区间为  $A_2, A_3$

即:  $A_1 = A_3, A_3 = A_4, F_3 = F_4$

$A_4 = A_1 + 0.618(A_2 - A_1), F_4 = f(A_4)$

重复(3), 继续进行比较。

以上比较直到  $|(F_3 - F_4)/F_3| < E$ , 且

$|(A_4 - A_3)/A_3| \leq E$  为止。

此时  $A_3, A_4$  小者为极小点, 相应函数值为极小值。

直接优选法要求预先确定寻查区间  $[A_1, A_2]$ , 这是由进退算法来确定的。进退算法确定寻查区间要预先给一个初始点  $A$  和步长  $H$ 。进退算法如下:

前进算法: 先计算  $f(A)$  和  $f(A+H)$ ,

若  $f(A) > f(A+H)$ , 则步长加大 2 倍, 计算  $f(A+3H)$ , 若  $f(A+H) \leq f(A+3H)$ , 则  $A_1 = A, A_2 = A+3H$ ; 否则, 将步长再加倍, 并重复前述运算。

后退运算: 若  $f(A) < f(A+H)$ , 则将步长缩为原来的  $1/4$ , 并改变符号, 即将步长改为  $-\frac{H}{4}$ 。若  $f(A - \frac{H}{4}) > f(A)$ , 则  $A_1 = A - \frac{H}{4}, A_2 = A+H$ ; 否则将步长加倍并继续后退。

## 二、程序说明

本程序由两部分组成。第一部分从 8002 句到 8056 句, 是用进退算法来确定寻查区间。第二部分从 8060 句到 8112 句, 是用 0.618 法寻查极小值。程序中第一条语句 8000 为要寻查的函数。如果要寻查的不是极小值而是极大值, 则在第一句定义函数时应在目标函数前加一负号, 最后打印出来的极值点即为所寻求的极大值点, 但打印出的函数值要改变符号后, 才为所求的极大值。

本程序没有使用数组, 全部使用简单变量, 分别为:

$A$ : 寻优区间起始点。

$H$ : 进退算法确定寻查区间的初始步长。

$E$ : 函数最优值的误差精度。

$A_1, A_2$ : 寻优区间。

$A_3, A_4$ : 按 0.618 法选取的试点。

$F_1 \cdots F_4$ : 分别为  $A_1 \cdots A_4$  各点函数值。

## 三、程序使用

1. 由 BASIC 引入本程序, 修改 8000 句, 定义用户自己的寻优函数  $FNC(X)$ 。

2. 用键盘命令  $RUN$  启动本程序, 机器自动印出: "INPUT A, H, E" 询问这三个值。当使用者用键盘回答以上三个值后, 机器立即印出寻优区间  $A_1, A_2$  之值, 然后印出 "ANS:" 字样, 接着印出的就是极值点和极值。

## 四、试题

$$f(x) = \frac{\sin x}{x}$$

极小点  $X^* = 4.49341$

极小值  $f(X^*) = -0.217234$

## 五、程序清单和运行结果

程序清单

```

8000 DEF FNC(X)=SIN(X)/X
8002 PRINT "INPUT A,H,E"
8004 PRINT
8006 INPUT A,H,E
8008 LET A1=A
8010 LET A2=A+H
8012 LET F1=FNC(A1)
8014 LET F2=FNC(A2)
8016 IF F1>F2 THEN GOTO 8034
8018 LET H=-H/4
8020 LET A1=A1+H
8022 LET F2=F1
8024 LET F1=FNC(A1)
8026 IF F2<F1 THEN GOTO 8046
8028 LET A2=A1-H
8030 LET H=2*H
8032 GOTO 8020
8034 LET H=2*H
8036 LET A2=A2+H
8038 LET F1=F2
8040 LET F2=FNC(A2)
8042 IF F2<F1 THEN GOTO 8052
8044 PRINT
8046 PRINT A1,A2
8048 PRINT
8050 GOTO 8056
8052 LET A1=A2-H
8054 GOTO 8034
8056 PRINT "ANS:"
8058 PRINT
8060 LET T=0.618
8062 LET A3=A2-T*(A2--A1)
8064 LET F3=FNC(A3)
8066 LET A4=A1+T*(A2-A1)
8068 LET F4=FNC(A4)
8070 IF ABS((F4-F3)/F3)<E THEN GOTO 8086
8072 IF F3>F4 THEN GOTO 8094
8074 LET A2=A4
8076 LET A4=A3
8078 LET F4=F3

```

```

8080 LET A3=A2-T*(A2-A1)
8082 LET F3=FNC(A3)
8084 GOTO 8070
8086 IF ABS((A4-A3)/A3)=<E THEN GOTO 8102
8088 LET A1=A3
8090 LET A2=A4
8092 GOTO 8062
8094 LET A1=A3
8096 LET A3=A4
8098 LET F3=F4
8100 GOTO 8066
8102 IF F3<F4 THEN GOTO 8108
8104 PRINT A4,F4
8106 GOTO 8112
8108 PRINT A3,F3
8110 GOTO 8112
8112 END

```

第一次运行结果

```

RUN
INPUT A,H,E
? 1 ? .01 ? .001
2.27      6.11
ANS,
4.49456   -.217233
END AT 8112

```

第二次运行结果

```

RUN
INPUT A,H,E
? 4 ? .01 ? .001
4.31      5.27
ANS,
4.49393   -.217234
END AT 8112

```

第三次运行结果

```

RUN
INPUT A,H,E
? 4 ? .01 ? .0001
4.31      5.27
ANS,
4.4935    -.217234

```

## §2 二次插值法求极值

## 一、算法概要

多项式是逼近函数的一种常用工具。在寻求函数极小点的区间（即寻查区间）上，我们完全可以利用已知若干点处的函数值来构成一个低次插值多项式，用它作为求极小点的函数的近似表达式，并用这个多项式的极小点作为原函数的极小点的近似。低次多项式的极小点比较容易计算。例如，设 $P(X)$ 为函数 $f(x)$ 的一个插值多项式，多项式 $P(x)$ 的极值点就是方程：

$P'(x)=0$  的根。我们只要求出这个方程的根并加以判断，就可以得到函数  $f(x)$  极小点的近似位置。重复使用此法，直到得出符合要求的结果为止。

常用的插值多项式 $P(x)$ 为二次或三次。一般说来三次插值公式的收敛性好一些，但在导数不便计算时，三点二次插值也是常用的一种有效办法。

假定目标函数 $f(x)$ 在三点 $x_1 < x_2 < x_3$ 的函数值分别为 $f_1, f_2$ 和 $f_3$ ，可以利用这三点及相应的函数值作二次插值公式，即令

$$P(x) = a_0 + a_1x + a_2x^2 \quad (16-2-1)$$

为所需的插值多项式，它应满足条件：

$$\begin{cases} P(x_1) = a_0 + a_1x_1 + a_2x_1^2 = f_1 \\ P(x_2) = a_0 + a_1x_2 + a_2x_2^2 = f_2 \\ P(x_3) = a_0 + a_1x_3 + a_2x_3^2 = f_3 \end{cases} \quad (16-2-2)$$

对多项式(16-2-1)求导数并令其等于零有：

$$\begin{aligned} P'(x) &= a_1 + 2a_2x = 0 \\ \therefore x &= -a_1/2a_2 \end{aligned} \quad (16-2-3)$$

式(16-2-3)就是计算近似极小点的公式。为确定这个极小点，只需从(16-2-2)中解出 $a_1$ 和 $a_2$ 来：

在方程组(16-2-2)中，利用相邻两个方程消去 $a_0$ ，得 $a_1, a_2$ 的方程组为：

$$\begin{cases} a_1(x_1 - x_2) + a_2(x_1^2 - x_2^2) = f_1 - f_2 \\ a_1(x_2 - x_3) + a_2(x_2^2 - x_3^2) = f_2 - f_3 \end{cases}$$

则

$$\begin{cases} a_1 = \frac{(x_2^2 - x_3^2)f_1 + (x_3^2 - x_1^2)f_2 + (x_1^2 - x_2^2)f_3}{(x_1 - x_2)(x_2 - x_3)(x_3 - x_1)} \\ a_2 = -\frac{(x_2 - x_3)f_1 + (x_3 - x_1)f_2 + (x_1 - x_2)f_3}{(x_1 - x_2)(x_2 - x_3)(x_3 - x_1)} \end{cases}$$

代入(16-2-3)式，便得到极值点公式：

$$x = \frac{1}{2} \frac{(x_2^2 - x_3^2)f_1 + (x_3^2 - x_1^2)f_2 + (x_1^2 - x_2^2)f_3}{(x_2 - x_3)f_1 + (x_3 - x_1)f_2 + (x_1 - x_2)f_3}$$

$$\text{令 } c_1 = (f_3 - f_1)/(x_3 - x_1)$$

$$c_2 = ((f_2 - f_1)/(x_2 - x_1) - C_1)/(x_2 - x_3)$$

则可以证明，极小点为：

$$x = \frac{1}{2} (x_1 + x_3 - C_1/C_2) \quad (16-2-4)$$



式 (16-2-4) 就是本程序所使用的计算公式。

## 二、程序说明

本程序由两部分构成。第一部分是确定寻查区间。二次插值区间要求  $x_1 < x_2 < x_3$  或者  $x_1 > x_2 > x_3$ ；而相应的函数值要求  $F_1 > F_2, F_3 > F_2$ ，即中间那一点的函数值比另外两点的函数值要小。第二部分为二次插值法求极小值。即假定  $(x_1, F_1)$ 、 $(x_2, F_2)$  和  $(x_3, F_3)$  为二次函数上的三个已知点，来求此二次函数的极小点  $x_4$  和极值  $F_4$ 。如果  $|F_2 - F_4|$  足够小，则认为二者之中的小者为极值点；否则再以  $x_2, x_4$  和  $x_1, x_3$  中的一个（视  $F_2, F_4$  哪个大而定），形成二次函数上的三个新的已知点，再求新的极值点，如此迭代循环。

整个程序框图如下：

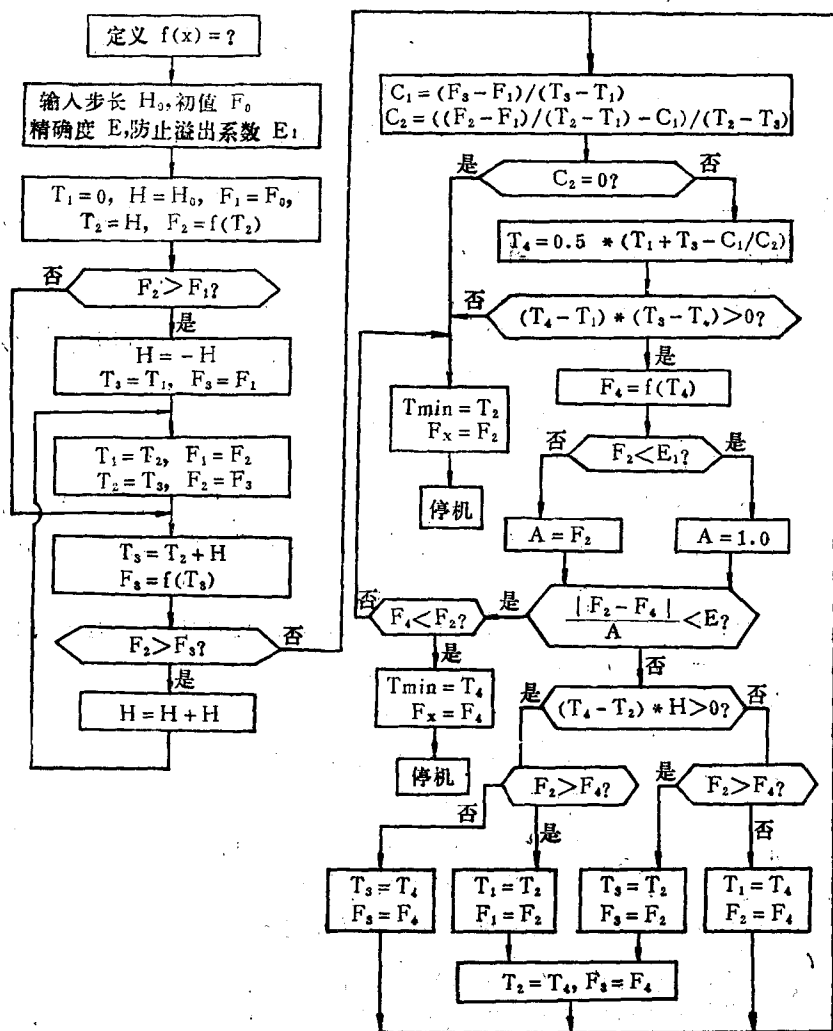


图 16-2-1

## 三、程序的使用

1. 由BASIC解释程序引入本程序，修改8000句，使之变成用户的寻优函数。
2. 用键盘命令RUN启动本程序运行，机器自动印出：“INPUT H, F0, E, E1” 询

问寻查的步距 $H_0$ ，函数初值 $F_0$ ，以及精度 $E$ 和防止溢出系数 $E_1$ 。

3. 用键盘回答以上各值后，立即印出求出的极值点和相应的函数值。

#### 四、试题：

$$f(x) = \sin(x)/x$$

$$\text{极值点 } x^* = 4.49341$$

$$f(x^*) = -0.217234$$

#### 五、程序清单及运行结果

##### 程序清单

```
8000 DEF FNC(X)=SIN(X)/X
8002 PRINT "INPUT H0,F0,E,E1"
8004 INPUT H0,F0,E,E1
8006 PRINT
8008 LET T1=0
8010 LET H=H0
8012 LET F1=F0
8014 LET T2=H
8016 LET F2=FNC(T2)
8018 IF F2>F1 THEN GOTO 8038
8020 LET T3=T2+H
8022 LET F3=FNC(T3)
8024 IF F2>F3 THEN GOTO 8054
8026 LET C1=(F3-F1)/(T3-T1)
8028 LET C2=((F2-F1)/(T2-T1)-C1)/(T2-T3)
8030 IF C2=0 THEN GOTO 8110
8032 LET T4=0.5*(T1+T3-C1/C2)
8034 IF (T4-T1)*(T3-T4)>0 THEN GOTO 8058
8036 GOTO 8110
8038 LET H=-H
8040 LET T3=T1
8042 LET F3=F1
8044 LET T1=T2
8046 LET F1=F2
8048 LET T2=T3
8050 LET F2=F3
8052 GOTO 8020
8054 LET H=H+H
8056 GOTO 8044
8058 LET F4=FNC(T4)
8060 IF F2<E1 THEN GOTO 8076
8062 LET A=F2
```

```

8064 IF ABS(F2-F4)/A<E THEN GOTO 8080
8066 IF (T4-T2)*H>0 THEN GOTO 8086
8068 IF F2>F4 THEN GOTO 8104
8070 LET T1=T4
8072 LET F1=F4
8074 GOTO 8026
8076 LET A=1
8078 GOTO 8064
8080 IF F2>F4 THEN GOTO 8110
8082 PRINT T4,F4
8084 GOTO 8114
8086 IF F2>F4 THEN GOTO 8094
8088 LET T3=T4
8090 LET F3=F4
8092 GOTO 8026
8094 LET T1=T2
8096 LET F1=F2
8098 LET T2=T4
8100 LET F2=F4
8102 GOTO 8026
8104 LET T3=T2
8106 LET F3=F2
8108 GOTO 8098
8110 PRINT T2,F2
8112 END

```

第一次运行结果:

RUN

INPUT H<sub>0</sub>,F<sub>0</sub>,E,E<sub>1</sub>

? 0.01 ? 5 ? 0.1 ? 0.001

4.49366            - .217234

END AT 8112

第二次运行结果:

RUN

INPUT H<sub>0</sub>,F<sub>0</sub>,E,E<sub>1</sub>

? 0.1 ? 20 ? 0.1 ? 0.001

4.4938            - .217234

END AT 8112

### §3 有理插值求极值

#### 一、算法概要

除了可以用多项式逼近函数外，用有理多项式——连分式也可以很好地逼近函数。有理插值求极值就是用有理连分式逼近目标函数，求连分式的极值来代替求原目标函数的极值。

若已知一函数  $f(x)$  在基点  $x_0, x_1, \dots, x_n$  处的值为  $y_0, y_1, \dots, y_n$ ，则用连分式近似表示如下，连分式形式为：

$$\varphi(x) = a_0 + \frac{x-x_0}{a_1 + \frac{x-x_1}{a_2 + \dots a_{n-1} + \frac{x-x_{n-1}}{a_n}}} \quad (16-3-1)$$

《16-3-1》式可以改写为：

$$\left\{ \begin{array}{l} \varphi(x) = \varphi_0(x) = a_0 + \frac{x-x_0}{\varphi_1(x)} \\ \varphi_1(x) = a_1 + \frac{x-x_1}{\varphi_2(x)} \\ \dots\dots\dots \\ \varphi_{n-1}(x) = a_{n-1} + \frac{x-x_{n-1}}{\varphi_n(x)} \\ \varphi_n(x) = a_n \end{array} \right. \quad (16-3-2)$$

$\because \varphi(x_i) = f(x_i) = y_i, i=0, 1, \dots, n$ ，代入 (16-3-2) 式可得，

$$\left\{ \begin{array}{l} a_0 = \varphi_0(x_0) = y_0 \\ \varphi_0(x_i) = y_i \\ \varphi_{i+1}(x_i) = \frac{x_i - x_i}{\varphi_i(x_i) - a_i}, j=0, 1, \dots, i-1 \\ a_i = \varphi_i(x_i), i=1, 2, \dots, n \end{array} \right. \quad (16-3-3)$$

这样确定了逼近函数  $\varphi(x)$

将上述逼近方法用于  $z=f'(x)$  的反函数  $x=\psi(z)$ ，假定  $x=x_i$  时  $f'(x_i)=z_i$ ，即  $x_i=\psi(z_i)$

《 $i=0, 1, \dots, n$ 》，这时  $\psi(z)$  可以近似表示为：

$$\begin{aligned} x &= \psi(z) \\ &= b_0 + \frac{z-z_0}{b_1 + \frac{z-z_1}{b_2 + \frac{z-z_2}{\dots b_{n-1} + \frac{z-z_{n-1}}{b_n}}}} \end{aligned} \quad (16-3-4)$$

同样 (16-3-4) 式也可以改写为：

$$\psi(z) = \psi_0(z) = b_0 + \frac{z-z_0}{\psi_1(z)}$$

$$\psi_1(z) = b_1 + \frac{z - z_1}{\psi_2(z)}$$

.....

$$\psi_{n-1}(z) = b_{n-1} + \frac{z - z_{n-1}}{\psi_n(z)}$$

$$\psi_n(z) = b_n$$

同样计算  $b_0, b_1, \dots, b_n$  的递推公式为.

$$\begin{cases} b_0 = \psi_0(z_0) = x_0 \\ \psi_0(z_i) = x_i \\ \psi_{i+1}(z_i) = \frac{z_i - z_1}{\psi_i(z_i) - b_i}, \quad j=0, \dots, i-1 \\ b_i = \psi_i(z_i), \quad i=1, 2, \dots, n \end{cases} \quad (16-3-5)$$

则可以导出极值点的坐标  $x^*$

$\therefore x^*$  对应于  $z=0$

$\therefore$  由 (16-3-4) 式得出

$$X^* = b_0 - z_0 / (b_1 - z_1 / (b_2 - \dots - z_{n-1} / b_n) \dots) \quad (16-3-6)$$

$n-2$  个

以下问题就转入如何求出  $z_i = f'(x_i)$ 。一种办法就是直接求  $f'(x)$  并算出  $f'(x_i)$ ；另一种办法是用差商来代替微商。前一种算法有时由于  $f'(x)$  的表达式很繁不易求出而很难实现，后一种办法由于进行一维寻查时每计算一个导数值都需要多次计算函数值而花费较多的机器时间，因此，下面我们推荐一种算法。

对 (16-3-2) 式进行求导数得：

$$\begin{cases} \varphi_0'(x) = \frac{\varphi_1(x) - (x - x_0)\varphi_1'(x)}{\varphi_1^2(x)} \\ \varphi_1'(x) = \frac{\varphi_2(x) - (x - x_1)\varphi_2'(x)}{\varphi_2^2(x)} \\ \dots\dots\dots \\ \varphi_{n-1}'(x) = \frac{\varphi_n(x) - (x - x_{n-1})\varphi_n'(x)}{\varphi_n^2(x)} \\ \varphi_n'(x) = 0 \end{cases} \quad (16-3-7)$$

由 (16-3-7) 式可以求出  $z$  来。

$$z_0 = \varphi_0'(x_0) = \frac{1}{\varphi_1(x_0)}$$

$$\varphi_1'(x_1) = \frac{1}{\varphi_2(x_1)}$$

$$z_1 = \varphi_0'(x_1) = \frac{\varphi_1(x_1) - (x_1 - x_0)\varphi_1'(x_1)}{\varphi_1^2(x_1)}$$

$\therefore$  一般有：

$$\varphi_i'(x_i) = \frac{1}{\varphi_{i+1}(x_i)}$$

$$\varphi_i'(x_i) = \frac{\varphi_{i+1}(x_i) - (x_i - x_i)\varphi_{i+1}'(x_i)}{\varphi_{i+1}^2(x_i)}$$

$$j = i-1, \dots, 0$$

$$z_i = \varphi_0'(x_i)$$

式中用到的 $\varphi_i(x_i)$ 可按下列公式计算:

$$\varphi_0(x_i) = a_0$$

$$\varphi_i(x_i) = a_i + \frac{x_i - x_i}{\varphi_{i+1}(x_i)}$$

## 二. 程序说明

1. 程序使用数组为X, A, Z, B, F. 使用简单变量为X<sub>0</sub>, H, N, U, W.
2. 程序中第一语句为目标函数.

## 三. 程序使用

1. 由 BASIC 解释程序引入本程序. 用自定义函数语句在标号为8090语句中定义用户目标函数FNY(X), 然后由键盘命令RUN启动本程序. 机器自动印出:  
"INPUT X<sub>0</sub>, H, N"

询问插值基点初始值X<sub>0</sub>和步长H, (本算法对步长取法不严格, 尽可能小些即可) 以及迭代次数初值N<sub>0</sub>. 每次从X<sub>0</sub>按步长H取8点插值基值. 按上述算法求出X\*. 然后将X\*作为X<sub>0</sub>重新按步长H又取8点, 反复计算, 每计算一次使N≤N+1. 一直继续到N=7为止.

2. 由键盘输入以上三个值, 机器自动印出每次计算的X\*值.

## 四. 试题:

$$f(x) = \sin(x)/x$$

$$\text{极值点: } x^* = 4.49341$$

## 五. 程序清单及运行结果

### 程序清单

```

8000 DEF FNY(X)=SIN(X)/X
8002 DIM X(8), A(8), Z(8), B(8), F(8,8)
8004 GOTO 8024
8006 REM SUBROUTINE
8008 LET F6=A(6)+(U-X(6))/A(7)
8010 LET F5=A(5)+(U-X(5))/F6
8012 LET F4=A(4)+(U-X(4))/F5
8014 LET F3=A(3)+(U-X(3))/F4
8016 LET F2=A(2)+(U-X(2))/F3
8018 LET F1=A(1)+(U-X(1))/F2
8020 LET F0=A(0)+(U-X(0))/F1
8022 RETURN
8024 PRINT "INPUT X0, H, N"
8026 INPUT X0, H, N
8028 PRINT
8030 FOR I=0 TO 7
8032 LET X(I)=X0+I*H
8034 LET F(0, I)=FNY(X(I))

```

```

8036 NEXT I
8038 PRINT
8040 LET A(0)=F(0,0)
8042 FOR J=1 TO 7
8044   FOR I=J TO 7
8046     LET F(J,I)=(X(I)-X(J-1))/(F(J-1,I)-A(J-1))
8048   NEXT I
8050   LET A(J)=F(J,J)
8052 NEXT J
8054 FOR I=0 TO 7
8056   LET U=X(I)
8058   GOSUB 8006
8060   LET F(7,I)=0
8062   LET F(6,I)=1/A(7)
8064   LET F(5,I)=(F6-(U-X(5))*F(6,I))/(F6*F6)
8066   LET F(4,I)=(F5-(U-X(4))*F(5,I))/(F5*F5)
8068   LET F(3,I)=(F4-(U-X(3))*F(4,I))/(F4*F4)
8070   LET F(2,I)=(F3-(U-X(2))*F(3,I))/(F3*F3)
8072   LET F(1,I)=(F2-(U-X(1))*F(2,I))/(F2*F2)
8074   LET Z(I)=(F1-(U-X(0))*F(1,I))/(F1*F1)
8076   LET F(0,I)=X(I)
8078 NEXT I
8080 LET B(0)=X(0)
8082 FOR J=1 TO 7
8084   FOR I=J TO 7
8086     LET F(J,I)=(Z(I)-Z(J-1))/(F(J-1,I)-B(J-1))
8088   NEXT I
8090   LET B(J)=F(J,J)
8092 NEXT J
8094 LET M=B(3)-Z(3)/(B(4)-Z(4)/(B(5)-Z(5)/(B(6)-Z(6)/B(7))))
8096 LET W=B(0)-Z(0)/(B(1)-Z(1)/(B(2)-Z(2)/M))
8098 PRINT W
8100 LET N=N+1
8102 IF N=7 THEN GOTO 8108
8104 LET X0=W
8106 GOTO 8030
8108 END

```

第一次运行结果:

RUN

INPUT X0, H, N

? 3.14159 ? 0.01 ? 3

4.46237

4.49343

4.49334

4.49341

END AT 8108

第二次运行结果:

RUN

INPUT X0, H, N

? 3.0 ? 0.01 ? 6

3.85954

END AT 8108

第三次运行结果:

RUN

INPUT X0, H, N

? 4.0 ? 0.01 ? 5

4.4693

4.49343

END AT 8108

运行结果说明: 一般初始基值点离最优点较远时, N值输入要小一点, 这样可以多迭代几次, 结果就精确多了, 如第一次运行。如第二次运行, 初始点离最优点又远, 迭代次数又只有一次, 所以精度下降很多。

#### 本章参考文献

南京大学数学系编

《最优化方法》



## 第十七章 线性规划、动态规划

## §1 非全人工变数单纯形法解线性规划问题

本程序采用非全人工变数单纯形法，解一般线性规划问题。使用本程序可以给出线性规划问题的目标函数的极小值或极大值，及其相对应的变量的取值。对于不可行的线性规划问题，或者无界的线性规划问题，本程序都给出相应的信息。

### 一、计算方法概要

所谓单纯形法就是：线性规划的每一个可行解是由一组基所组成的，从最初的一组基开始，通过将这组基线性相关的其他变量进行替换，从而产生一组新的基，这组新基导致得到一个新的较好的目标函数可行解。这样反复进行下去，直到求得最优解为止。

有时, 根据已知的约束条件, 找不出明显的基, 这时需人为地引进一些变量(松弛变量), 也即人工变数, 以便得出最初的一组基, 为了缩短计算过程, 我们只对部分约束条件加入人工变数, 而对另一些条件则不加人工变数, 也即非全人工变数。

本程序运用上述方法来求解线性规划问题。线性规划的求解方法，主要归结为解决两个问题。其一是：有了一组基以后，在有可行解的情况下，如何选择一个与这组基线性相关的变量，进行替换，以最快速度，趋向于最优解。同时，对其他变量也要作相应的变换；其二是如何确定目标函数是否已达到最优解（极大值或极小值），以及如何区别线性规划不可行及无界。

下面我们用算式来说明以上问题。

给定目标函数为:

$$f(x) = C_1 x_1 + C_2 x_2 + \dots + C_n x_n \quad (17-1-1)$$

给定约束条件为:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ \quad \quad \quad \dots \quad \quad \quad \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \end{cases} \quad (17-1-2)$$

$$\text{且 } x_i \geq 0$$

若要把以上约束条件写成等式，即如下：

[illegible]

其中  $x_{n+1}, x_{n+2}, \dots, x_{n+m}$  即为引进的人工变量。

式 (17-1-3) 也可写成如下形式:

$$p_1x_1 + p_2x_2 + \cdots + p_nx_n + p_{n+1}x_{n+1} + \cdots + p_{n+m}x_{n+m} = p_0 \quad (17-1-3)$$

其中:

$$p_i = \begin{pmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{mi} \end{pmatrix} \quad i \leq n$$

$$p_{n+j} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \leftarrow \text{第 } j \text{ 位} \quad j \leq m \text{ (这时令 (17-1-3) 式中的 } a_{in+j} = 1)$$

$$p_0 = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

对于 (17-1-3) 式存在一组基称为 B, 则令:

$$B = (p(1)p(2)p(3)\cdots p(m)) \quad p(i) \neq p_i$$

那么 (17-1-3) 式中的任一个矢量  $p_k, k=1, \dots, n+m$  都可由 B 来线性表出, 即

$$p_k = Y(1)_k p(1) + Y(2)_k p(2) + \cdots + Y(m)_k p(m) = BY_k$$

其中

$$Y_k = \begin{pmatrix} Y(1)_k \\ Y(2)_k \\ \vdots \\ Y(m)_k \end{pmatrix} \quad k=1, \dots, n+m \quad (17-1-4)$$

(17-1-4) 式又可以写成:

$$Y_k = B^{-1}P_k \quad k=1, 2, \dots, n+m \quad (17-1-5)$$

由线性变换, 式 (17-1-3) 可写成:

$$\theta(1)P(1) + \theta(2)P(2) + \cdots + \theta(m)P(m) = P_0 \quad (17-1-6)$$

其中  $P(i)$  为新基 B 的各个分量, 而令

$$\theta(i) = X(i) \quad i=1, 2, \dots, m \text{ 是变量矢量在某组基下形成的各分量。}$$

下面我们来推导如何找出替换变量  $\theta_k'$ , 产生一个新基。

在式 (17-1-6) 两边分别加上  $\theta_k' P_k$  一项得:

$$\begin{aligned} & \theta(1)P(1) + \theta(2)P(2) + \cdots + \theta(m)P(m) + \theta_k' P_k \\ &= P_0 + \theta_k' P_k \\ &= P_0 + \theta_k' Y(1)_k P(1) + \theta_k' Y(2)_k P(2) + \cdots + \theta_k' Y(m)_k P(m) \end{aligned} \quad (17-1-7)$$

式 (17-1-7) 经整理后得:

$$P_0 = (\theta(1) - \theta_k' Y(1)_k) P(1) + (\theta(2) - \theta_k' Y(2)_k) P(2) + \cdots + (\theta(m) - \theta_k' Y(m)_k) P(m) + \theta_k' P_k \quad (17-1-8)$$

在 (17-1-8) 式中, 右边有  $m+1$  项分量, 而初始基 B 只有  $m$  项分量, 所以有一项应该消去。这样可以获得一组新基。我们选择某一项趋近零最快的进行变换, 即取

$$\theta_k' = \frac{\theta(1)}{Y(1)_k} = \min \left\{ \frac{\theta(i)}{Y(i)_k} \right\}, \quad Y(i)_k > 0, \quad i=1, 2, \dots, m \quad (17-1-9)$$

若将第 1 个分量消去, 其他分量也应进行相应的变换, 其变换公式为:

$$\theta'(i) = \theta(i) - \frac{\theta(1)}{Y(1)_k} \cdot Y(i)_k, \quad i=1, 2, \dots, m \quad \text{且 } i \neq 1 \quad (17-1-10)$$

这样初始基 B 中的  $P(1)$  分量将由  $P_k$  来替换, B 中的其他分量也相应有所变化, 从而得到一个新基。

现在再来看看, 新基的变换, 对目标函数有什么影响。

在初始基下, 目标函数表示为:

$$f(x) = C(1)\theta(1) + C(2)\theta(2) + \dots + C(m)\theta(m)$$

若  $P_i$  (列向量) 是一个经过替换后新基的一个分量, 则由 (17-1-8) 式可以得到:

$$\theta'(1) = \theta(1) - \theta'_i y(1)_i, \quad \theta'(2) = \theta(2) - \theta'_i y(2)_i, \quad \dots$$

而在新基下的目标函数为:

$$\begin{aligned} f'(x) &= C(1)(\theta(1) - \theta'_i Y(1)_i) + C(2)(\theta(2) - \theta'_i Y(2)_i) + \dots + C(m)(\theta(m) - \theta'_i Y(m)_i) + C_i \theta'_i \\ &= C(1)\theta(1) + C(2)\theta(2) + \dots + C(m)\theta(m) - C(1)Y(1)_i \theta'_i - C(2)Y(2)_i \theta'_i - \dots - C(m)Y(m)_i \theta'_i + C_i \theta'_i \\ &= f(x) - \theta'_i (C(1)Y(1)_i + C(2)Y(2)_i + \dots + C(m)Y(m)_i - C_i) \end{aligned}$$

令  $Z_i = C(1)Y(1)_i + C(2)Y(2)_i + \dots + C(m)Y(m)_i$ , 则上式化为:

$$f'(x) = f(x) - \theta'_i (Z_i - C_i) \quad (17-1-11)$$

对于 (17-1-11) 式来说, 其中  $\theta'_i \geq 0$  ( $\because \theta'_i = X(j)$ )。在求极大值的情况下,  $Z_i - C_i$  必须为负值;  $Z_i - C_i$  为负值时,  $P_i$  才可以作为一个新基的分量而引入, 否则  $Z_i - C_i$  为正值,  $P_i$  就不可以引入。因为引入  $P_i$  的目的就是使  $f'(x)$  取值增大而趋于极大值。若所有引入的人工变数都为 0, 则对应的基底就是最优解, 即使  $Z_i - C_i$  无负值为止, 此时目标函数取极大值。

同样, 在求极小值的情况下, 要求  $Z_i - C_i$  必须取正值。当变换  $Z_i - C_i$  都不为正时, 目标函数取极小值。

由 (17-1-9) 式和 (17-1-10) 式知道, 基的变换是在  $Y(i)_k > 0$  的情况下进行的。若  $Y(i)_k < 0$ , 则变换的  $\theta(i)$  值将越来越大, 最终使目标函数趋于无穷。若  $Y(i)_k = 0$  时, 则会使  $\theta'_k$  取任意值, 引出无穷多组解, 即线性规划问题不可行。

非全人工变数单纯形法就是在出现约束条件有 “ $\geq$ ” 符号时, 引入人工变数, 而 “ $\leq$ ” 或 “ $=$ ” 的约束条件时, 则不引入人工变数。

例如有如下的约束条件:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n \leq b_2 \\ \dots \\ a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + \dots + a_{in}x_n \leq b_i \\ a_{i+1,1}x_1 + a_{i+1,2}x_2 + a_{i+1,3}x_3 + \dots + a_{i+1,n}x_n \geq b_{i+1} \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \dots + a_{mn}x_n \geq b_m \end{cases}$$

其中 1, 2, ..., i 项是含有  $\leq$  符号的约束条件; 而自 i 到 m 项约束条件是含有  $\geq$  符号。当把约束条件化为等式处理时, 则 i 项以前各个约束条件要引入一个带 + 号变量 (松弛变量), 而 i 项以后各约束条件要引入一个带 - 号的变量, 即为:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + x_{n+1} = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n + x_{n+2} = b_2 \\ \dots \\ a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n + x_{n+i} = b_i \\ a_{i+1,1}x_1 + a_{i+1,2}x_2 + \dots + a_{i+1,n}x_n - x_{n+i+1} = b_{i+1} \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n - x_{n+m} = b_m \end{cases}$$

若目标函数为  $f(x) = C_1X_1 + C_2X_2 + \dots + C_nX_n$  则可写成 A 矩阵为:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} & 0 & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & 0 & b_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{in} & 0 & b_i \\ a_{i+1,1} & a_{i+1,2} & \dots & a_{i+1,n} & -1 & b_{i+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} & -1 & b_m \\ C_1 & C_2 & \dots & C_n & 0 & 0 \\ Z_1 & Z_2 & \dots & Z_n & Z_{n+1} & Z_{n+2} \end{pmatrix} \quad (17-1-12)$$

在 A 阵中第 m+1 行是目标函数的系数, 第 m+2 行是引进的人工变量, 且有:

$$Z_n = -\sum_{i=1}^m a_{in}, \quad Z_{n+2} = -\sum_{i=1}^m b_i, \quad Z_{n+1} = -\sum_{i=1}^m (-1)$$

## 二、程序说明

本程序使用以下数组:

A(M+2, M+N+2) 为 (17-1-12) 所示的原始信息阵。

U(M+2, M+2) 为单位矩阵, 变换时, A 阵不变而改变 U 的元素内容。

X(M+2, M+2) 相当于 Y(i)<sub>k</sub>, 其中 X(M+2) 存放解。

S(M+2), 存放中间运算单元。

本程序中使用 P, M, N<sub>1</sub>, I, J, K 等简单变量, 其中 N<sub>1</sub> 为变量个数, M 为约束条件数。

## 三、上机操作步骤

1. 由 BASIC 引入本程序, 再用 RUN 键盘命令启动本程序执行, 机器将自动地印出:  
“NUMBER OF VARIABLES”, 询问要解的线性规划问题中变量的数目。
2. 使用者由键盘回答变量个数后, 机器又自动印出: “NUMBER OF CONSTRAINTS”, 询问线性规划问题中约束条件个数。
3. 使用者由键盘回答约束条件个数后, 机器又自动印出: “ENTER MATRIX A”, 要求使用者从键盘敲入 A 阵的各个元素之值。A 阵各元素要求按 (17-1-12) 式给的 A 阵表达式按行输入。

4. 等A阵输入完, 机器又自动印出:  
"ENTER OBJECTIVE FUNCTION", 要求使用者用键盘敲入目标函数中的各个系数值。
5. 等目标函数的系数值输入完, 机器又自动印出: "EVALUATION OF B(I)", 要求使用者用键盘敲入数组B(I)之值。具体规定如下: 由第一个约束条件开始, 若约束条件是 $\geq$ 号或 $=$ 号, 敲入数值1; 若约束条件为 $\leq$ 号, 则敲入数值0。
6. 当B(I)之值输入完, 机器转入计算。计算完将按不同情况给出计算的结果信息。
  - I 线性规划问题不可行, 则印出:  
"INFEASIBLE", 然后停在 7248。
  - II 线性规划问题无界, 则印出:  
"UNBOUND", 然后停在 7252。
  - III 解可行, 求极小值时印出:  
"OBJECTIVE FUNCTION IS MIN. = \* \* \* \* \*"  
并印出各个变量取值。
  - IV 解可行, 求极大值时印出:  
"OBJECTIVE FUNCTION IS MAX. = \* \* \* \* \*"  
并印出各个变量取值。

#### 四、例题

约束条件为:

$$\begin{cases} 4x_1 + 5x_2 + x_3 \geq 10 \\ 5x_1 + 2x_2 - x_3 \leq 10 \\ 3x_1 + 8x_2 + 2x_3 \leq 12 \\ -x_1 + x_2 + 4x_3 \geq 1 \end{cases}$$

目标函数:

$$f(x) = 5x_1 + 3x_2 + 2x_3 = \text{MIN}$$

结果:  $f(x) = 10.5882$

$$x_1 = 1.76471$$

$$x_2 = 0.588236$$

$$x_3 = 0$$

#### 五、程序清单及运行结果

```

7000 REM "THE PROGRAM OF LINEAR PROGRAMMING"
7002 REM "SOLUTIONS IN SIMPLEX METHOD"
7004 PRINT "NUMBER OF VARIABLES"
7006 INPUT N1
7008 PRINT
7010 PRINT "NUMBER OF CONSTRAINTS"
7012 INPUT M
7014 PRINT
7016 DIM A(M+2, 2 * N1+2), X(M+2, M+2), U(M+2, M+2), S(2 * N1+2),

```

```

      B(2 * N1+2)
7018 PRINT "ENTER MATRIX A"
7020 FOR I=1 TO M
7022   FOR J=1 TO N1+2
7024     INPUT A (I,J)
7026   NEXT J
7028 PRINT
7030 NEXT I
7032 PRINT "ENTER OBJECTIVE FUNCTION"
7034 FOR I=1 TO N1+2
7036   INPUT A (M+1,I)
7038 NEXT I
7040 PRINT
7042 LET F=1
7044 LET P=M+1
7046 PRINT
7048 PRINT "EVALUATION OF B (I) "
7050 LET R=0
7052 FOR I=1 TO M
7054   INPUT B (I)
7056   LET R=R+B (I)
7058 NEXT I
7060 PRINT
7062 LET N=N1+1
7064 IF R=0 THEN GOTO 7120
7066 FOR I=1 TO M+2
7068   LET S (I) =0
7070   LET S (I) =A(I,N1+2)
7072   LET A (I,N1+2) =0
7074 NEXT I
7076 LET R=0
7078 FOR I=1 TO M
7080   IF A (I,N1+1) ><-1 THEN GOTO 7090
7082   LET R=R+1
7084   IF R=1 THEN GOTO 7090
7086   LET A (1,N1+R) =A(I,N1+1)
7088   LET A (I,N1+1) =0
7090 NEXT I
7092 LET N=N1+R
7094 FOR I=1 TO M+1

```

```

7096 LET A(I,N+1)=S(I)
7098 NEXT I
7100 IF R=0 THEN GOTO 7106
7102 LET F=0
7104 LET P=M+2
7106 FOR J=1 TO N+1
7108 LET A(P,J)=0
7110 FOR I=1 TO M
7112 IF B(I) ><1 THEN GOTO 7116
7114 LET A(P,J)=A(P,J)-A(I,J)
7116 NEXT I
7118 NEXT J
7120 FOR I=1 TO P
7122 LET U(I,I)=1
7124 FOR J=1 TO P
7126 IF J=I THEN GOTO 7130
7128 LET U(I,J)=0
7130 NEXT J
7132 NEXT I
7134 LET E=0.000001
7136 LET W=-1
7138 FOR I=1 TO P
7140 LET B(I) =N+I
7142 NEXT I
7144 FOR I=1 TO P
7146 LET X(I) =0
7148 FOR J=1 TO P
7150 LET X(I) =X(I)+U(I,J) * A(J,N+1)
7152 NEXT J
7154 NEXT I
7156 IF F=1 THEN GOTO 7164
7158 IF X(P)+E<0 THEN GOTO 7166
7160 LET P=P-1
7162 LET F=1
7164 LET W=0
7166 LET K=0
7168 LET R=0
7170 FOR J=1 TO N
7172 LET S(J) =0
7174 FOR I=1 TO P

```

```

7176     LET S(J)=S(J)+U(P,I) * A(I,J)
7178     NEXT I
7180     IF S(J)-R+E=>0 THEN GOTO 7186
7182     LET R=S(J)
7184     LET K=J
7186     NEXT J
7188     IF K=0 THEN GOTO 7244
7190     LET R=1E+31
7192     LET L=0
7194     FOR I=1 TO P
7196         LET X(I,K) =0
7198         FOR J=1 TO P
7200             LET X(I,K)=X(I,K)+U(I,J) * A(J,K)
7202         NEXT J
7204         IF X(I,K)-E=<0 THEN GOTO 7218
7206         IF X(I)-E=<0 THEN GOTO 7218
7208         LET S(I)=0
7210         LET S(I)=X(I)/X(I,K)
7212         IF S(I)-R=>0 THEN GOTO 7218
7214         LET R=S(I)
7216         LET L=I
7218     NEXT I
7220     IF L=0 THEN GOTO 7250
7222     FOR J=1 TO P
7224         LET U(L,J)=U(L,J)/X(L,K)
7226     NEXT J
7228     FOR I=1 TO P
7230         IF I=L THEN GOTO 7238
7232         FOR J=1 TO P
7234             LET U(I,J)=U(I,J)-U(L,J) * X(I,K)
7236         NEXT J
7238     NEXT I
7240     LET B(L)=K
7242     GOTO 7144
7244     IF F=1 THEN GOTO 7254
7246     PRINT "INFEASIBLE"
7248     STOP
7250     PRINT "UNBOUND"
7252     STOP
7254     LET Y=-X(M+1)

```



```

7256 IF Y>0 THEN GOTO 7262
7258 PRINT "OBJECTIVE FUNCTION IS MAX.=";-Y
7260 GOTO 7264
7262 PRINT "OBJECTIVE FUNCTION IS MIN.=";Y
7264 FOR I=1 TO M
7266   LET S (I) =0
7268 NEXT I
7270 FOR I=1 TO M
7272   LET J=B (I)
7274   LET S (J) =X (I)
7276 NEXT I
7278 FOR I=1 TO N1
7280   PRINT "X (";I;" ) =" ;S (I)
7282   PRINT
7284 NEXT I
7286 END

```

运行结果

RUN

NUMBER OF VARIABLES

? 3

NUMBER OF CONSTRAINTS

? 4

ENTER MATRIX A

? 4 ? 5 ? 1 ? -1 ? 10

? 5 ? 2 ? -1 ? 0 ? 10

? 3 ? 8 ? 2 ? 0 ? 12

? -1 ? 5 ? 4 ? -1 ? 1

ENTER OBJECTIVE FUNCTION

? 5 ? 3 ? 2 ? 0 ? 0

EVALUATION OF B(I)

? 1 ? 0 ? 0 ? 1

OBJECTIVE FUNCTION IS MIN.=10.5882

X ( 1 ) =1.76471

X ( 2 ) =.588235

X ( 3 ) = 0

END AT 7286

#### 参考文献

[ 1 ] An-min Chung.《Linear Programming》 May 1962

[ 2 ] 《常用算法》 科学出版社

## §2 一维动态规划

### 一、方法概述:

如果有一个过程(如资源的分配、结构的设计)需要我们选择决策,使之目标函数取极值(如达到最大的经济效益;最小消耗量等)这类问题通称为规划问题。若目标函数是变量的线性函数,那就是人们所熟悉的线性规划问题。若目标函数是变量的非线性函数,或不连续的函数,甚至写不出函数的表达式。对这类问题的寻优,动态规划提供了一个有效的数值解法。

动态规划方法早年由Bellman提出的,他给出一个“最优决策”准则,虽然至今没有严格证明,但在实际应用中已由大量事实证明其正确性。动态规划的基本思想是将全过程分解成一系列简单的决策步骤(一系列的子过程)。对一个简单决策步骤逐步判定最优决策,最后形成全过程的最优决策。

现在假定有一个过程被我们分成 $N$ 个简单决策步骤。

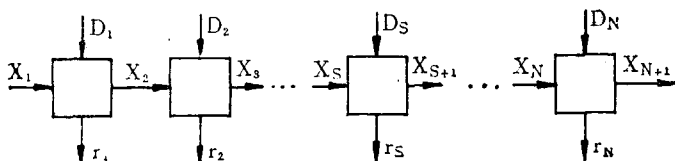


图 17-2-1 动态规划示意图

动态规划顺序方向 ← ————  
———→ 原过程顺序方向

其中:  $X_s$ ——第 $S$ 个决策步骤的输入状态变量。  
 $X_{s+1}$ ——第 $S+1$ 个决策步骤的输出状态变量。  
 $r_s$ ——第 $S$ 个决策步骤的收益函数。  
 $D_s$ ——第 $S$ 个决策步骤的控制量。

显然,  $r_s$ 是  $X_s$  和  $D_s$  的函数, 即  $r_s = R_s(X_s, D_s)$ 。假定全过程的总收益是各个步骤的收益的和, 记为  $\Gamma$ , 则

$$\Gamma = \sum_{s=1}^N R_s(X_s, D_s) \text{ 称为目标函数。}$$

在动态规划问题中, 就是希望  $\Gamma$  取极小值记为  $f_N$ , 则有:

$$f_N = \min_{\substack{X_1, \dots, X_N \\ D_1, \dots, D_N}} \Gamma = \min_{\substack{X_1, \dots, X_N \\ D_1, \dots, D_N}} \left[ \sum_{s=1}^N R_s(X_s, D_s) \right] \quad (17-2-1)$$

问题提法: 即选择一系列最优决策参数  $D_1, D_2, \dots, D_N$ , 从而有最优状态变量, 并使全过程的总收益为最小。

(17-2-1) 式是动态规划问题的数学描述, 直接方法求 (17-2-1) 式是无从下手的, 为此使用 (17-2-2) 式公式。可以将 (17-2-1) 式导出一个著名的动态规划的递推公式。

公式 (17-2-2) 如下: 即对任何实函数  $g(X_1)$ ,  $h(X_1, X_2)$  总有:

$$\min_{X_1, X_2} [g(X_1) + h(X_1, X_2)] = \min_{X_1} [g(X_1) + \min_{X_2} h(X_1, X_2)] \quad (17-2-2)$$

对 (17-2-1) 式反复使用 (17-2-2) 式的结果就可以得出 (17-2-3) 式如下:

$$\left. \begin{aligned} f_N &= \min_{\substack{X_N \\ D_N}} [R_N(X_N, D_N) + f_{N-1}] \\ f_{N-1} &= \min_{\substack{X_{N-1} \\ D_{N-1}}} [R_{N-1}(X_{N-1}, D_{N-1}) + f_{N-2}] \\ &\vdots \\ f_s &= \min_{\substack{X_s \\ D_s}} [R_s(X_s, D_s) + f_{s-1}] \\ &\vdots \\ f_2 &= \min_{\substack{X_2 \\ D_2}} [R_2(X_2, D_2) + f_1] \\ f_1 &= \min_{\substack{X_1 \\ D_1}} [R_1(X_1, D_1)] \end{aligned} \right\} \quad (17-2-3)$$

(17-2-3) 式就是求解动态规划问题的著名的递推公式。它的意义在于把全过程最优决策问题化为  $N$  个简单决策步骤的寻优问题，从  $S = 1$  到  $S = N$  即可求出全过程的最优决策。

Bellman 对一般动态规划问题给出了一个优化准则，即：“一个过程的最优决策，具有如下性质，即无论其初始状态和初始决策如何，其今后的决策的选择，对以前一决策所形成的状态为初始状态的过程而言，必须构成最优决策。”

在实际的问题中，通常是对目标函数在变量容许范围内按离散间隔取值进行优化，得到相对优化结果。在这种情况下，采用所谓列表计算方法求解是很方便可行的。即把每一简单决策步骤的最优决策、目标函数值列出来。如对第  $s$  步骤，假设  $X_{s-1}$  可能有  $K$  个值  $X_{s1}, X_{s2}, \dots, X_{sK}$ ，而控制量(决策)  $D_s$  可能取  $J$  个值  $D_{s1}, D_{s2}, \dots, D_{sj}$ 。对每个输入状态变量取值  $X_{si}$  将会有  $J$  个  $r_s$  值与  $f_{s-1}$  相配合，在这  $J$  个值中必有最优者记为  $f_{si}$ ，相应的决策参量值记为  $D_{spi}$ ，把它们列成表，则第  $S$  步所列的表如下：

状态变量 $X_s$ 的可能值	相应最优决策变量	相应的收益值
$X_{s1}$	$D_{sp1}$	$f_{s1}$
$\vdots$	$\vdots$	$\vdots$
$X_{si}$	$D_{spi}$	$f_{si}$
$\vdots$	$\vdots$	$\vdots$
$X_{sK}$	$D_{spK}$	$f_{sK}$

将所有  $N$  个步骤列出这些表，最后可以求出全过程的目标函数最优值。再按表反查回去得到一系列最优决策参数  $D_{s, op}$ ，最优状态变量  $X_{s, op}$  ( $s = 1, 2, \dots, N$ )。以上计算工作可以人工进行，也可以使用计算机。

为了更加具体地说明动态规划的计算过程,我们举例来加以说明。

例:考虑非线性控制系统,其差分方程给定如下:

$$X(K+1) = X(K) + [X^2(K) + u(K)] \cdot \Delta$$

其中 $X(K)$ 是一维状态变量, $u(K)$ 是一维控制变量。

我们的问题是求最优决策(控制) $u(n)$

满足约束条件 $|u(n)| \leq 1$

$$\text{使目标函数 } J = \sum_{i=1}^N |X(i) - u^3(i)| \cdot \Delta = \min$$

其中 $\Delta$ ——系统采样间隔时间。这里取 $\Delta = 0.1$ 秒。

为了不使叙述繁琐,我们考虑 $N = 3$ 的情况,并简化实际问题,取状态变量 $|X(K)| = 3$ 。

∴考虑以下离散值

$$X(K) = \{-3, -2, -1, 0, 1, 2, 3\}$$

$$u(K) = \{-1, 0, 1\}$$

按照前面所讲的动态规划计算过程,则从第一个决策步骤开始,一个步骤一个步骤进行。每个步骤先取一个状态离散值,即令 $X(K) = -3$ ,分别取 $u(K) = -1(0, 1)$ 等决策量,代入系统的差分方程,则分别可以求出下一个状态变量的值,若是不合法的,则丢掉不算。若是合法的,就可以代入收益函数式中,求出一个收益值。若不是开始的第一步,还要加上本步骤以前的累积收益值,得到本步骤的累积收益值。在这几个不同决策值的计算结果中比较累积收益值,取最小的那一组数据,即相应的状态离散值,最佳决策值和最小累积收益函数值。

然后再换一个系统状态变量的离散值 $X(K) = -2$ ,重复以上计算比较过程,找到另外一组数据值。用简单示意图表示如下:

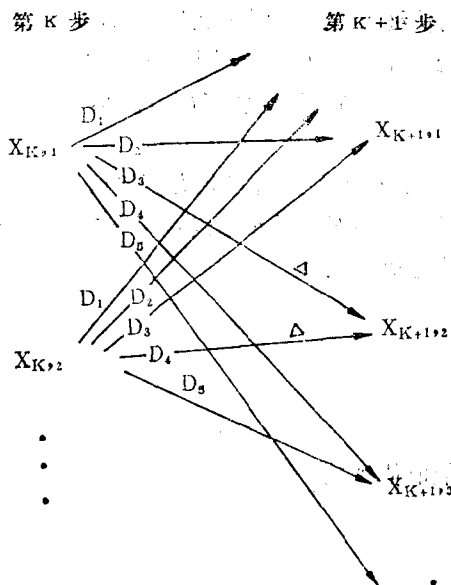


图17-2-2 动态规划计算步骤示意图

最小的那个“历程”,因此,有一个再次选优的问题。如果从第 $K$ 步的不同状态离散值,采用

对第 $K$ 步的 $X_{K,1}$ 状态离散值,取不同的决策量,经过系统的差分方程计算,若不是不合理的,就一定是第 $K+1$ 步中的某几个不同的状态离散值,则通过比较累积收益函数值后,可以选出一个最佳的决策值,图中用 $\Delta$ 标出。

同样,对第 $K$ 步的 $X_{K,2}$ 状态离散值,取不同的决策量,经过系统的差分方程计算,若不是不合理的,则也一定为第 $K+1$ 步中某几个不同的离散值。通过比较累积收益函数值后,也可以选出一个最佳决策值,图中用 $\Delta$ 标出。从图中看出,虽然第 $K$ 步状态离散值不同,最佳决策选出的不一样,但第 $K+1$ 步的状态离散值相同。从系统的物理过程来看,相当于系统经过不同的“历程”后,到此时刻系统达到的某一个相同的状态。从最优的角度来看,是要选其中累积收益函数值

不同的控制量后,第 $K+1$ 步的状态离散值不同,则说明到此时刻为止,系统仍然经历不同的“历程”之中,还很难说那一个“历程”的累积收益函数值最小,因此还不能进行比较而取舍,只好留待下一步再比较。以上计算过程,用人工列表计算如下:

第 $K$ 步状态离散值	-3			-2			-1			0			1			2			3		
决策量	-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
第 $K+1$ 步状态离散值	-2	-2	-2	-2	-2	-2	0	0	0	-1	-1	-1	1	1	1	2	2	3	4	4	4
累积收益值	0.2	0.3	0.4	0.1	0.2	0.3	0	0.1	0.2	0.1	0	0.1	0.2	0.1	0	0.3	0.2	0.1	/	/	/

从上表数据看出,每一个状态离散值都挑出一个最佳决策值,表中用\*标明的就是。从表中看出,第一个状态离散值和第二个状态离散值,它们的下一状态的取值都是-2,则它们两者之间还要比较,选收益值最小的,则取 $X(1)=-2$ 的那一组。另外, $X(1)=3$ ,由状态方程求出的下一状态离散值为4,是不合理的甩掉,其它的都是合法的,且取下一状态的离散值都是各不相同的,因此不能再比较了。

到此为止,计算过程的第一步骤才算完成,如法炮制,分别再计算第二步、第三步的。动态规划的计算过程就是如此。

动态规划计算过程只是动态规划方法的第一部分,还有第二个部分。第一个部份是数据准备阶段。在控制系统中应用时,这一部分工作是可以“离线”工作的。第二部分是最佳轨线追踪过程,这一部分工作在自动控制系统中应用时一般是“在线”执行的。由追踪的方向不同,有“前向”动态规划和“后向”动态规划。我们这里讲的,以及后面给出的程序和运行的例题都是属于“后向”动态规划,即追踪的方向从终点到起始时刻倒推过来的。这就要求在追踪开始,给出终点时刻(第 $N$ 步)和终点状态离散值,然后找到 $N-1$ 时刻的最优状态离散值 $X(N-1)$ 和最优决策值 $U(N-1)$ ,直到起始时刻的最优状态离散值 $X(1)$ 和最优决策值 $U(1)$ ,使得累积收益函数值最小。

后向动态规划也有只给出追踪终点时刻值(第 $N$ 步),而不给出终点状态离散值。这时除了首先要比较终点时刻取那一个离散值时累积收益函数最小而取最小的终点状态离散值外,其它都是一样的。对于本程序而言,必须给出终点状态离散值。

## 二、程序说明

程序中使用以下简单变量:

$N$ ——子过程(或决策步骤)的个数

$K$ ——每个子过程中状态可能取值的个数。

$J$ ——每个子过程中控制量(决策量)可能取值个数。

$N_1$ ——追踪终点时刻。

同时使用以下数组

$X(I)$ ——每个子过程中状态变量可能取的离散值,  $I=1, \dots, K$ 。

$U(I)$ ——每个子过程中控制量(决策)可能取的离散值,  $I=1, \dots, J$ 。

$S(I)$ ——追踪后的最优状态,  $I=1, \dots, N_1$ 。

$D(I)$ ——追踪后的最优决策,  $I=1, \dots, N_1$ 。

$R(I)$ ——追踪后的每个子过程的累积收益函数值。

为了便于后向追踪，在程序中使用了数组  $F(I, J), I=1, \dots, N, J=1, \dots, 3K$ ，它用于存放计算过程中所得到的中间信息，其每个元素定义如下：

$F(I+1, 3 * R_s - 2)$  ——下一状态的离散值。

$F(I, 3 * R_s - 1)$  ——本状态的最佳决策值。

$F(I, 3 * R_s)$  ——本状态的最小收益值。其中  $R_s = 1, \dots, K$

### 三、使用操作

1. 由BASIC程序输入本程序，由RUN/键盘命令启动后，机器自动印出：  
"INPUT NOM. OF STEP"  
询问总的步数N。
2. 由键盘回答总的步数后，机器又自动印出："INPUT K, X(K), J AND U(J) IN THE STEP" 询问每一步中状态变量离散值的个数K和具体值X(K)以及每一步中决策变量离散值个数J和具体值U(J)。
3. 由键盘回答以上各个值后，机器自动印出  $F(I, J)$  的中间信息值，三个值一行，F阵中一行印成一块，作为使用者检查验证使用。
4. 计算机完成计算过程后，开始追踪工作。所以印出："BEGIN TO FIND THE OPTIMAL TRAIL"  
"INPUT STATE X AND STEP  $N_1$  VALUE"  
询问追踪终点时刻  $N_1$  和状态变量离散值  $X(N_1)$ 。
5. 由键盘回答后，若  $N_1 > N$ ，非法，印出 "THIS  $N_1$  VALUE IS ILLEGAL" 并要求重新回答。若与终点状态变量离散值X相应的累积收益值为  $1E+20$ ，则判断出这个状态值是不可控的，印出 "THIS X VALUE IS OUT OF CONTROL" 要求重新输入。否则进行追踪，并印出追踪结果来。
6. 用户使用时只要改8005和8010句，分别定义状态方程和收益函数即可，其他语句不用动。

### 四、例题

已知系统离散状态方程为

$$X(K+1) = X(K) + [X^2(K) + u(K)] \cdot \Delta$$

约束条件为

$$|X(K)| \leq 3 \quad |u(K)| \leq 1$$

$$J = \sum_{i=1}^N |X(i) - u^s(i)| \cdot \Delta = \min$$

取  $N = 3$ ， $\Delta = 0.1$  秒（采样间隔时间）

### 五、程序清单和运行结果

#### 程序清单

```
8005 DEF FNR(X)=ABS (X-Q*Q*Q)*.1
8010 DEF FNX(X)=X+(X*X+Q)*.1
8015 PRINT "INPUT NOM. OF STEP"
8020 INPUT N
8025 PRINT
```

```

8030 PRINT "INPUT K,X(K),J AND U(J) IN THE STEP"
8035 INPUT K
8040 PRINT
8045 DIM X(K),F(N+1,3*K+3)
8050 FOR I=1 TO K
8055     INPUT X(I)
8060 NEXT I
8065 PRINT
8070 INPUT J
8075 PRINT
8080 DIM U(J),R(N),D(N),V(N)
8085 FOR I=1 TO J
8090     INPUT U(I)
8095 NEXT I
8100 PRINT
8105 FOR I=1 TO N
8110     FOR I1=3 TO 3*K STEP 3
8115         LET F(I,I1)=1E+20
8120     NEXT I1
8125 NEXT I
8130 FOR I1=1 TO N
8140     FOR I2=1 TO K
8150         LET S=X(I2)
8160         FOR I3=1 TO J
8170             LET Q=U(I3)
8180             LET X1=FNX(S)
8190             IF X1<0 GOTO 8220
8200             LET X1=INT (X1+.5)
8210             GOTO 8230
8220             LET X1=-INT (ABS (X1)+.5)
8230             FOR L=1 TO K
8240                 IF X1=X(L) GOTO 8270
8250             NEXT L
8260             GOTO 8360
8270             LET R1=FNR(S)
8280             IF I1=1 GOTO 8300
8290             LET R1=R1+F(I1-1,3*I2)
8300             IF I3=1 GOTO 8320
8310             IF R1>R2 GOTO 8360
8320             LET R2=R1

```

```

8330         LET R3=Q
8340         LET R4=S
8350         LET R5=L
8360     NEXT I3
8370     IF R2>F(I1,3 * R5) GOTO 8410
8380     LET F (I1+1,3 * R5-2)=R4
8390     LET F(I1,3 * R5-1)=R3
8400     LET F(I1,3 * R5)=R2
8410 NEXT I2
8420 NEXT I1
8440 FOR I=1 TO N
8450     FOR L=0 TO K-1
8460         PRINT F(I,3 * L+1),F(I,3 * L+2),F(I,3 * L+3)
8470     NEXT L
8480     PRINT
8490 NEXT I
8500 PRINT
8520 PRINT "BEGIN TO FIND THE OPTIMAL TRAIL"
8530 PRINT
8540 PRINT "INPUT STATE X AND STEP N1 VALUE"
8550 INPUT S,N1
8555 PRINT
8556 IF N1<=N GOTO 8560
8558 PRINT "THIS N1 VALUE IS ILLEGAL"
8559 GOTO 8540
8560 FOR L=1 TO K
8570     IF S=X(L) GOTO 8586
8580 NEXT L
8582 PRINT "THIS X VALUE IS ILLEGAL"
8584 GOTO 8540
8586 IF F(N1,3 * L)=1E+20 GOTO 8590
8588 GOTO 8610
8590 PRINT "THIS X VALUE IS OUT OF CONTROL"
8600 GOTO 8540
8610 PRINT "THE OPTIMAL TRAIL"
8620 PRINT
8630 PRINT TAB (5); "STEP", "STATE", "DECIDE", "GAIN"
8640 PRINT
8650 FOR I=N1 TO 1 STEP -1
8660 LET V(I)=X(L)

```



```

8670 LET D(I)=F(I, 3*L-1)
8680 LET R(I)=F(I,3*L)
8685 IF I=1 GOTO 8730
8688 LET R4=F(I,3*L-2)
8690 FOR L=1 TO K
8700 IF R4=X(L) GOTO 8720
8710 NEXT L
8712 PRINT "HAVE ERR"
8715 STOP
8720 NEXT I
8730 FOR I=1 TO N1
8740 PRINT TAB (5);I,V(I),D(I),R(I)
8750 NEXT I
8760 END

```

# 运行结果

RUN

INPUT NOM. OF STEP

? 3

INPUT K, X(K), J AND U(J) IN THE STEP

? 7

? - 3? - 2? - 1? 0? 1? 2? 3

? 3

? - 1? 0? 1

0	0	1E+20
0	- 1	.1
0	- 1	0
0	0	0
0	1	0
0	1	.1
0	0	1E+20
0	0	1E+20
- 2	- 1	.2
- 1	- 1	0
0	0	0
1	1	0
2	1	.2
0	0	1E+20

0	0	1E+20
-2	-1	.3
-1	-1	0
0	0	0
1	1	0
2	1	.3
0	0	1E+20

BEGIN TO FIND THE OPTIMAL TRAIL  
INPUT STATE X AND STEP N1 VALUE

? 3 ? 3

THIS X VALUE IS OUT OF CONTROL  
INPUT STATE X AND STEP N1 VALUE

? .5 ? 2

THIS X VALUE IS ILLEGAL  
INPUT STATE X AND STEP N1 VALUE

? 2 ? 4

THIS N1 VALUE IS ILLEGAL  
INPUT STATE X AND STEP N1 VALUE

? -2 ? 3

THE OPTIMAL TRAIL

STEP	STATE	DECIDE	GAIN
1	-2	-1	.1
2	-2	-1	.2
3	-2	-1	.3

END AT 8760

# 第十八章 控制系统的计算机辅助设计

## §1 画控制系统传递函数的 NYQUIST 图

### 一、基本概念

无论是进行控制系统分析还是进行控制系统的综合，NYQUIST 图都是十分有用的工具。

NYQUIST 图就是控制系统传递函数  $G(j\omega)$  的极坐标图。它是当  $\omega$  由零变化到无穷大时，表示在极坐标上的  $G(j\omega)$  幅值与  $G(j\omega)$  相角的关系图。在极坐标图上正（负）相角是从正实轴开始以反（顺）时针旋转来定义的。

设传递函数具有如下形式：

$$G(S) = \frac{K(1+T_a S)(1+T_b S) \cdots}{S^\lambda (1+T_1 S)(1+T_2 S) \cdots}$$

$$= \frac{b_0 + b_1 S + \cdots + b_m S^m}{a_0 + a_1 S + \cdots + a_n S^n} \quad (n > m) \quad (18-1-1)$$

则相应的 NYQUIST 图的形状有以下几种

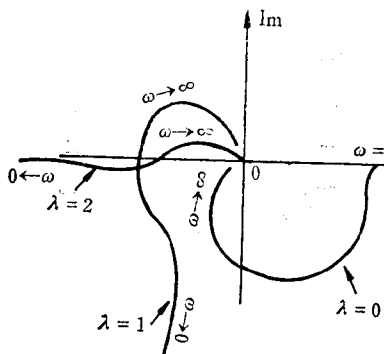


图 18-1-1 NYQUIST 图几种类型

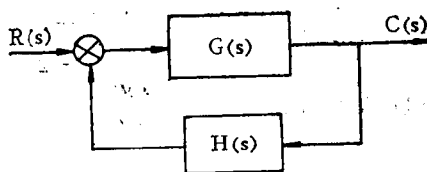


图 18-1-2 系统结构图

从图 (18-1-1) 中看出，若传递函数的分母多项式的阶次高于分子多项式的阶次，那  $G(j\omega)$  的轨迹将以顺时针方向收敛于原点，在  $\omega = \infty$  处，其轨迹与实轴或虚轴相切。

当  $G(j\omega)$  的分母多项式阶次与分子多项式相同时，则极坐标图起自实轴上某一有限远点，且止于实轴上有限远点。

应用 NYQUIST 图，可以分析控制系统稳定性。对于如下图所示的系统而言，可能有三种情况。

(1) 不包围  $-1 + j0$  点。如果这时  $G(s)H(s)$  在右半  $s$  平面上没有极点，则说明系统是稳定的，否则系统不稳定。

(2) 反时针包围  $-1 + j0$  点。这时如果反时针包围的次数等于  $G(s)H(s)$  在右半  $s$  平面上的极点数，则系统是稳定的，否则，系统是不稳定的。

(3) 顺时针包围  $-1+j0$  点。这时系统是不稳定的。

应用 NYQUIST 图, 可以给出相位裕量  $\gamma$ 。 $\gamma$  之值一方面可以表明系统是不是稳定, 另一方面能指出不稳定系统应如何加以改动而变成稳定系统。所谓相位裕量就是指在开环传递函数的幅值  $|G(j\omega)|$  等于 1 时的频率 (增益交界频率) 上, 使系统达到不稳定边缘所需要的附加相位滞后量。即

$$\gamma = 180^\circ + \phi \quad (18-1-2)$$

其中  $\phi$  为开环传递函数在增益交界频率上的相角。为了使最小相位系统 ( $n > m$ ) 稳定, 相位裕量  $\gamma$  必须为正。

## 二、程序说明

本程序适用于  $G(s)$  的两种给出形式, 即多项式形式和按环节连乘形式, 分别为:

$$G(s) = \frac{b_0 + b_1 s + \dots + b_m s^m}{a_0 + a_1 s + \dots + a_n s^n} \quad (18-1-3)$$

或

$$G(s) = \frac{K(1+T_{11}s)(1+T_{12}s)\dots(1+T_{1m}s)}{(1+T_{21}s)(1+T_{22}s)\dots(1+T_{2n}s)} \quad (18-1-4)$$

画 NYQUIST 图其实就是使  $\omega$  变化, 求出  $G(j\omega)$  的实部  $R(\omega)$  和虚部  $I(\omega)$ 。

当  $G(s)$  用多项式方式给出时, 则令

$$G(j\omega) = \frac{\theta_1 + j\theta_2}{\theta_3 + j\theta_4}$$

其中  $\theta_1 = b_0 - b_2\omega^2 + b_4\omega^4 - \dots$

$$\theta_2 = b_1\omega - b_3\omega^3 + b_5\omega^5 - \dots$$

$$\theta_3 = a_0 - a_2\omega^2 + \dots$$

$$\theta_4 = a_1\omega - a_3\omega^3 + \dots$$

$$\begin{cases} R(\omega) = (\theta_1 * \theta_3 + \theta_2 * \theta_4) / (\theta_3^2 + \theta_4^2) \\ I(\omega) = (\theta_2 * \theta_3 - \theta_1 * \theta_4) / (\theta_3^2 + \theta_4^2) \end{cases} \quad (18-1-5)$$

当  $G(s)$  用环节连乘方式给出时, 可令各环节为如下形式:

$$G_1(j\omega) = \frac{A_{11} + jA_{12}}{A_{13} + jA_{14}} \quad (18-1-6)$$

则由 (18-1-6) 式可以推导出第一个环节的实部  $T_4$  和虚部  $T_5$  得:

$$T_4 = (A_{11} * A_{13} + A_{12} * A_{14}) / (A_{13}^2 + A_{14}^2)$$

$$T_5 = (A_{12} * A_{13} - A_{11} * A_{14}) / (A_{13}^2 + A_{14}^2)$$

而对于以后各环节将如法炮制得出实部  $T_7$  和虚部  $T_8$ :

$$T_7 = (A_{11} * A_{13} + A_{12} * A_{14}) / (A_{13}^2 + A_{14}^2)$$

$$T_8 = (A_{12} * A_{13} - A_{11} * A_{14}) / (A_{13}^2 + A_{14}^2)$$

由此可以导出两个环节连乘后的形式为

$$(T_4 + jT_5)(T_7 + jT_8) \quad (18-1-7)$$

由 (18-1-7) 式又得出两个环节连乘后的实部  $R(\omega)$  和  $I(\omega)$  有:

$$\begin{cases} R(\omega) = T_4 * T_7 - T_5 * T_8 \\ I(\omega) = T_4 * T_8 + T_5 * T_7 \end{cases} \quad (18-1-8)$$

由 (18-1-7) 式和 (18-1-8) 式, 可以求出所有环节连乘后的实部和虚部之值。

求增益交界频率和相角裕量使用以下公式。

根据以上求出的  $R(\omega)$  和  $I(\omega)$  之值, 找出满足

$$\sqrt{R^2(\omega) + I^2(\omega)} = \min \text{ 的 } \omega_1 \text{ 值,}$$

则 增益交界频率  $\omega_c = \omega_1$

$$\text{此时 } \phi = \tan^{-1} \left( \frac{I(\omega)}{R(\omega)} \right)$$

则 相角裕量为  $\gamma = 180^\circ + \phi$

本程序使用的频率  $\omega$  是从 0.01 变到  $10^4$ , 每一频率倍程上取 10 点。当  $|R(\omega)| + |I(\omega)| < 0.001$  时强迫结束计算。

### 三、程序使用

(1) 由 BASIC 引入本程序, 用键盘命令 RUN 启动本程序。机器自动印出:

"IS POLYNOMIAL FORM (M=0) OR UNIT FORM (M=1)?" 询问数据输入形式。若以多项式形式给出, 回答 0, 否则回答 1。

(2) 当使用者通过键盘回答 0 时, 机器又自动印出: "INPUT MAX. DEGREE OF POLYNOMIAL", 询问多项式的最大阶数 (以分母的阶数为准); 当使用者回答为 1 时, 机器自动印出: "INPUT NUMBER OF UNIT", 询问环节数。

(3) 使用者通过键盘回答以上问题后, 机器要求输入具体参数。待数据全部输入完毕, 转入计算, 每当频率  $\omega$  选择一个值 (0.01, 0.02, ..., 1, 2, ..., 10000) 就印出一对数值, 分别为  $R(\omega)$  和  $I(\omega)$  之值。

最后印出增益交界频率  $\omega_c$  和相角裕量。

### 四、程序清单和运行结果

#### 程序清单

```
8000 PRINT "IS POLYNOMIAL FORM (M=0) OR UNIT FORM(M=1)?"
8002 PRINT "INPUT M"
8004 INPUT M
8006 PRINT
8008 IF M=1 THEN GOTO 8160
8010 PRINT "INPUT MAX. DEGREE OF POLYNOMIAL"
8012 INPUT N
8014 PRINT
8016 DIM A(3,N+2), R(100), I(100)
8018 PRINT "INPUT COEFFI. OF POLY."
8020 FOR I=1 TO 2
8022   FOR J=0 TO N
8024     INPUT A(I, J)
8026   NEXT J
8028 PRINT
8030 NEXT I
8032 PRINT A(1,0);
8034 FOR I=1 TO N
```

```

8036 IF A(1, I)=0 THEN GOTO 8040
8038 PRINT "+", A(1, I), "S(", I, ")";
8040 NEXT I
8042 PRINT
8044 PRINT "....."
8046 PRINT A(2, 0);
8048 FOR I=1 TO N
8050 IF A(2, I)=0 THEN GOTO 8054
8052 PRINT "+", A(2, I), "S(", I, ")";
8054 NEXT I
8056 PRINT
8058 LET W=.01
8060 LET M=0
8062 FOR I1=0 TO 5
8064 FOR J=W * 10 ↑ I1 TO W * 10 ↑ (I1+1) STEP W * 10 ↑ I1
8066 LET M=M+1
8068 LET K=1
8070 LET Q1=A(1, 0)
8072 LET Q3=A(2, 0)
8074 FOR I2=2 TO INT(N/2) * 2 STEP 2
8076 LET K1=K
8078 GOSUB 8266
8080 LET Q1=Q1+K2 * A(1, I2) * J ↑ I2
8082 LET Q3=Q3+K2 * A(2, I2) * J ↑ I2
8084 LET K=K+1
8086 NEXT I2
8088 LET K=0
8090 LET Q2=0
8092 LET Q4=0
8094 FOR I2=1 TO INT(N/2) * 2+1 STEP 2
8096 LET K1=K
8098 GOSUB 8266
8100 LET Q2=Q2+K2 * A(1, I2) * J ↑ I2
8102 LET Q4=Q4+K2 * A(2, I2) * J ↑ I2
8104 LET K=K+1
8106 NEXT I2
8108 LET Q5=Q3 * Q3+Q4 * Q4
8110 LET R(M)=(Q1 * Q3+Q2 * Q4)/Q5
8112 LET I(M)=(Q2 * Q3-Q1 * Q4)/Q5
8114 IF ABS(R(M))+ABS(I(M))<.001 THEN GOTO 8128

```

```

8116      PRINT R(M), I(M);
8118      LET Q9=Q8
8120      LET Q8=ABS(SQR(R(M)*R(M)+I(M)*I(M))-1)
8122      IF Q8=>Q9 THEN GOTO 8128
8124      LET Q6=M
8126      LET Q7=J
8128      NEXT J
8130      PRINT
8132      NEXT I1
8134      PRINT "W(C)=", Q7
8136      LET Q8=ATN(I(Q6)/R(Q6))*57.3
8138      IF I(Q6)>0 THEN GOTO 8150
8140      IF R(Q6)>0 THEN GOTO 8146
8142      PRINT "R=", Q8
8144      GOTO 8284 ;
8146      PRINT "R=", 180+Q8
8148      GOTO 8284
8150      IF R(Q6)>0 THEN GOTO 8156
8152      PRINT "R=", Q8
8154      GOTO 8284
8156      PRINT "R=", 180+Q8
8158      GOTO 8284
8160      PRINT "INPUT NUMBER OF UNIT"
8162      INPUT N
8164      PRINT
8166      PRINT "INPUT CONSTANT OF UNIT"
8168      PRINT
8170      DIM A(N+1, 5), R(100), I(100)
8172      LET W=.01
8174      LET M=0
8176      FOR I=1 TO N
8178          FOR J=1 TO 4
8180              INPUT A(I, J)
8182          NEXT J
8184      PRINT
8186      NEXT I
8188      GOTO 8246
8190      FOR I1=0 TO 5
8192          FOR J=W*10↑I1 TO W*10↑(I1+1)STEP W*10↑I1
8194              LET M=M+1

```

```

8196     LET T3=A(1,3)*A(1,3)+*A(1,4)*A(1,4)*J*J
8198     LET T4=(A(1,1)*A(1,3)+A(1,2)*A(1,4)*J*J)/T3
8200     LET T5=(A(1,2)*A(1,3)-A(1,1)*A(1,4))*J/T3
8202     IF N=1 THEN GOTO 8220
8204     FOR I2=2 TO N
8206         LET T6=A(I2,3)*A(I2,3)+A(I2,4)*A(I2,4)*J*J
8208         LET T7=(A(I2,1)*A(I2,3)+A(I2,2)*A(I2,4)*J*J)/T6
8210         LET T8=(A(I2,2)*A(I2,3)-A(I2,1)*A(I2,4))*J/T6
8212         LET T9=T4
8214         LET T4=T4*T7-T5*T8
8216         LET T5=T9*T8+T5*T7
8218     NEXT I2
8220     LET R(M)=T4
8222     LET I(M)=T5
8224     IF ABS(R(M))+ABS(I(M))<0.01 THEN GOTO 8238
8226     PRINT R(M); I(M);
8228     LET Q9=Q8
8230     LET Q8=ABS(SQR(R(M)*R(M)+I(M)*I(M))-1)
8232     IF Q8=>Q9 THEN GOTO 8238
8234     LET Q6=M
8236     LET Q7=J
8238     NEXT J
8240     PRINT
8242     NEXT I1
8244     GOTO 8134
8246     FOR I=1 TO N
8248         PRINT "("; A(I, 1); "+"; A(I, 2); "S)";
8250     NEXT I
8252     PRINT
8254     PRINT "....."
8256     FOR I=1 TO N
8258         PRINT ")"; A(I, 3); "+"; A(I, 4); "S)";
8260     NEXT I
8262     PRINT
8264     GOTO 8190
8266     IF K1=0 THEN GOTO 8280
8268     LET K2=1
8270     FOR I3=1 TO K1
8272         LET K2=(-1)*K2
8274     NEXT I3

```



```

8276 LET K2=SGN(K2)
8278 RETURN
8280 LET K2=1
8282 RETURN
8284 GOTO 8286
8286 END

```

(1) 第一次运行结果

$$G(s) = \frac{10 + 4s + 0.7s^2}{1 + 3s + 5s^2 + 6s^3}$$

RUN

IS POLYNOMIAL FORM(M=0) OR UNIT FORM(M=1)?

INPUT M

? 0

INPUT MAX. DEGREE OF POLYNOMIAL

? 3

INPUT COEFFI. OF POLY.

? 10? 4? 0.7? 0

? 1? 3? 5? 6

10+4s(1)+0.7s(2)

.....  
1+3s(1)+5s(2)+6s(3)

9.99713 - .259984 9.98850 - .519872 9.97422 - .779571 9.95423 - 1.039  
9.92862 - 1.29807 9.89744 - 1.55673 9.86076 - 1.81493 9.81865 - 2.07264  
9.7712 - 2.32984

9.71847 - 2.58656 8.912 - 5.14927 7.4969 - 7.87765 4.65141 - 10.9776  
- 1.52978 - 12.59 - 7.36988 - 7.64307 - 6.59859 - 2.12226 - 4.37771  
- .117384 - 2.87471 0.397424

- 1.96804 .476001 - .222498 7.07784E-2 - 7.88643E-2 1.50353E-3  
- 4.04998E-2 - 1.18263E-2 - 2.47842E-2 - 1.43853E-2 - 1.67844E-2  
- 1.42444E-2 - 1.21427E-2 - 1.33838E-2 - 9.20309E-3 - 1.23805E-2  
- 1.2208E-3 - 1.14141E-2  
- 5.81952E-3 - 1.05367E-2

W(C)=2

R=-17.6476

END AT 8286

第二次运行结果

$$G(s) = \frac{20}{s(1+s)(1+0.5s)}$$

RUN

IS POLYNOMIAL FORM(M=0) OR UNIT FORM(M=1)?

INPUT M

```

? 1
INPUT NUMBER OF UNIT
? 3
INPUT CONSTANT OF UNIT
? 1 ? 0 ? 0 ? 1
? 1 ? 0 ? 1 ? 1
? 20 ? 0 ? 1 ? 0.5
(1+0s)(1+0s)(20+0s)
.....
(0+1s)(1+1s)(1+0.5s)
-29.9963 -1999.65 -29.985 -999.301 -29.9663 -665.618 -29.9401
-498.603 -29.9065 -398.255 -29.8655 -331.242 -29.8172 -283.278
-29.7616 -247.22 -29.6988 -219.1
-29.6289 -196.539 -28.5600 -93.2979 -26.9173 -57.1246 -24.8674
-38.1301 -22.5882 -26.353 -20.2375 -18.4386 -17.937 -12.8976
-15.7696 -8.93612 -13.7835 -6.07496
-12 -4.00003 -3.00001 .999998 -.923082 .717951 -.352943 .411766
-.159152 .244033 -8.10816E-2 .153154 -4.52833E-2 .101348 -2.71495E-2
7.01361E-2 -1.72168E-2 5.03749E-2
-1.14243E-2 3.73194E-2
W(c)=2.99999
R=-37.8776
END AT 8286

```

## §2 由系统的频率响应特性数据拟合系统的传递函数

### 一、方法简介:

不失一般性, 系统的传递函数可以写成:

$$Y(j\omega_k) = \frac{a_0 + a_1(j\omega_k) + a_2(j\omega_k)^2 + \dots + a_m(j\omega_k)^m}{1 + b_1(j\omega_k) + b_2(j\omega_k)^2 + \dots + b_n(j\omega_k)^n}$$

$$= \frac{N(j\omega_k)}{D(j\omega_k)} = \frac{\alpha + j\beta}{\sigma + j\tau} \quad n > m \quad (18-2-1)$$

其中  $\alpha = a_0 - a_2\omega_k^2 + a_4\omega_k^4 - a_6\omega_k^6 + \dots$

$\beta = a_1\omega_k - a_3\omega_k^3 + a_5\omega_k^5 - a_7\omega_k^7 + \dots$

$\sigma = 1 - b_2\omega_k^2 + b_4\omega_k^4 - b_6\omega_k^6 + \dots$

$\tau = b_1\omega_k - b_3\omega_k^3 + b_5\omega_k^5 - b_7\omega_k^7 + \dots \quad (18-2-2)$

在选定的频率  $\omega = \omega_k$ ,  $k=1, 2, \dots, p$  下, 测得系统频率响应数据(或由快速富里叶变换由系统的输入输出数据得出的系统频率响应数据)。

有:

$$Y^*(j\omega_k) = R_k^* + jI_k^*$$

令测量值与拟合值之差为:

$$\begin{aligned} \varepsilon_k &= Y(j\omega_k) - Y^*(j\omega_k) \\ &= \frac{N(j\omega_k)}{D(j\omega_k)} - Y^*(j\omega_k) \end{aligned} \quad (18-2-3)$$

为数学上处理方便，对误差进行加权处理，即乘以  $D(j\omega_k)$ ，则有：

$$\begin{aligned} e_k &= \varepsilon_k \cdot D(j\omega_k) = N(j\omega_k) - D(j\omega_k) \cdot Y^*(j\omega_k) \\ &= (\alpha + j\beta) - (\sigma + j\tau)(R_k^* + jI_k^*) \\ &= X_k + jY_k \end{aligned}$$

其中

$$\begin{aligned} X_k &= (\alpha - R_k^* \sigma + I_k^* \tau) \\ Y_k &= (\beta - R_k^* \tau - I_k^* \sigma) \end{aligned} \quad (18-2-4)$$

$\therefore$  得到加权的总均方差为：

$$E = \sum_{k=1}^p e_k \cdot e_k^* = \sum_{k=1}^p (X_k^2 + Y_k^2) \quad (18-2-5)$$

其中  $e_k^*$  为  $e_k$  的共轭复数。

将 (18-2-2) 式代入到 (18-2-4) 式，得到  $X_k$  和  $Y_k$  的展开式（以取  $W_k$  的最高次幂  $\leq 7$  为例）：

$$\begin{aligned} X_k &= (\alpha - R_k^* \sigma + I_k^* \tau) \\ &= (a_0 - a_2 \omega_k^2 + a_4 \omega_k^4 - a_6 \omega_k^6) - R_k^* (1 - b_2 \omega_k^2 + b_4 \omega_k^4 - b_6 \omega_k^6) + I_k^* (b_1 \omega_k - b_3 \omega_k^3 + b_5 \omega_k^5 - b_7 \omega_k^7) \\ &= a_0 - a_2 \omega_k^2 + a_4 \omega_k^4 - a_6 \omega_k^6 + b_1 I_k^* \omega_k + b_2 R_k^* \omega_k^2 - b_3 I_k^* \omega_k^3 - b_4 R_k^* \omega_k^4 + b_5 I_k^* \omega_k^5 + b_6 R_k^* \omega_k^6 - b_7 I_k^* \omega_k^7 - R_k^* \\ Y_k &= (\beta - R_k^* \tau - I_k^* \sigma) \\ &= (a_1 \omega_k - a_3 \omega_k^3 + a_5 \omega_k^5) - R_k^* (b_1 \omega_k - b_3 \omega_k^3 + b_5 \omega_k^5 - b_7 \omega_k^7) - I_k^* (1 - b_2 \omega_k^2 + b_4 \omega_k^4 - b_6 \omega_k^6) \\ &= a_1 \omega_k - a_3 \omega_k^3 + a_5 \omega_k^5 - b_1 R_k^* \omega_k + b_2 I_k^* \omega_k^2 + b_3 R_k^* \omega_k^3 - b_4 I_k^* \omega_k^4 - b_5 R_k^* \omega_k^5 + b_6 I_k^* \omega_k^6 + b_7 R_k^* \omega_k^7 - I_k^* \end{aligned} \quad (18-2-6)$$

将 (18-2-6) 式代入 (18-2-4) 式可知，加权的总均方差  $E$ ，不仅是量测值  $\omega_k$ ， $R_k^*$ ， $I_k^*$  的函数，而且也是待求系数  $a_i$  ( $i=0, 1, 2, \dots, m$ ) 和  $b_i$  ( $i=1, 2, \dots, n$ ) 的函数。所以为求拟合多项式的各个系数  $a_i$  和  $b_i$ ，分别求  $E$  对它们之中每一个的偏导数并让其等于零，则有：

$$\begin{aligned} \frac{\partial E}{\partial a_i} &= 0 \quad (i=0, \dots, m) \\ \frac{\partial E}{\partial b_j} &= 0 \quad (j=1, \dots, n) \end{aligned} \quad (18-2-7)$$

经整理后可得一组以待求系数为变量的线性方程组 ( $2 \times n$  个联立方程组)

$$AX=B \quad (18-2-8)$$

以  $m=6$ ， $n=7$  为例， $A$ 、 $X$  和  $B$  阵各元素分别为：



在方程 (18—2—8) 中, 各字母表示的定义式如下:

$$\begin{aligned}\lambda_i &= \sum_{k=1}^p \omega_k^{-1} \\ S_i &= \sum_{k=1}^p \omega_k^{-1} \cdot R_k \cdot \\ T_i &= \sum_{k=1}^p \omega_k^{-1} \cdot I_k \cdot \\ U_i &= \sum_{k=1}^p \omega_k^{-1} (R_k \cdot^2 + I_k \cdot^2)\end{aligned}\quad (18-2-9)$$

在推导 (18—2—8) 式的联立方程组中, 是将 (18—2—3) 式的  $\varepsilon_k$  乘以  $D(j\omega_k)$ , 得出加权总均方差, 这是此种方法的计算技巧所在。使用这种方法, 在较窄的频率段中 (低频较好), 求出的传递函数多项式系数精度在 0.001 以上, 基本上达到工程上的要求。但是这种方法有两个严重的弱点:

(1) 如果所给的系统的频率响应特性数据是跨几个频率倍程, 则在 A 阵中的各元中, 低频的值和高频的值比较起来非常小, 因此在低频段不能获得较好的拟合精度要求。

(2) 如果  $G(s)$  在复平面上有极点, 使  $|D(j\omega_k)|^2$  在实验频率点上取值范围变化很大, 因而引入较大的误差, 影响拟合的结果精度。为此作如下改进:

把 (18—2—4) 式改写成如下式:

$$\begin{aligned}e_k &= \frac{\varepsilon_k \cdot D(j\omega_k)_L}{D(j\omega_k)_{L-1}} \\ &= \frac{N(j\omega_k) - Y^*(j\omega_k) \cdot D(j\omega_k)_L}{D(j\omega_k)_{L-1}} \\ &= \frac{X_k + jY_k}{D(j\omega_k)_{L-1}}\end{aligned}\quad (18-2-10)$$

则:

$$E = \sum_{k=1}^p |e_k|^2 = \sum_{k=1}^p |X_k + jY_k|^2 / |D(j\omega_k)_{L-1}|^2 \quad (18-2-11)$$

此时联立方程组形式不变, 只不过 A 阵和 B 阵中各字母的定义式改变如下:

$$\begin{aligned}\lambda_i &= \sum_{k=1}^p \omega_k^{-1} \omega_{KL} \\ S_i &= \sum_{k=1}^p \omega_k^{-1} R_k \cdot \omega_{KL} \\ T_i &= \sum_{k=1}^p \omega_k^{-1} I_k \cdot \omega_{KL} \\ U_i &= \sum_{k=1}^p \omega_k^{-1} (R_k \cdot^2 + I_k \cdot^2) \omega_{KL}\end{aligned}\quad (18-2-12)$$

而  $\omega_{KL} = 1 / |D(j\omega_k)_{L-1}|^2$

由于  $D(j\omega_k)$  值初始时是不知道的, 设为 1。在迭代运算中, 总是用  $L-1$  次迭代中求出的  $b_{1L}, b_{2L}, \dots, b_{nL}$  来计算  $\omega_{kL}$  之值, 为第  $L$  次迭代做好准备。迭代直到前后两次加权值不变为止。

## 二、程序说明

本程序大致可分为以下几个部分

(I) 从 8000 句~8016 句为数据输入部分。

(II) 从 8018~8036 句, 是模拟产生一个系统的频率特性数据部分。产生出来的实频响应数据送  $R(I)$  数组; 产生的虚频响应数据送入  $I(I)$  数组。

(III) 从 8040~8144 句, 列主元消去法解线性方程组的专用子程序。

(IV) 从 8146~8218 句, 为一些加工子程序。

(V) 从 8228~8478 句, 为计算线性方程组的增广矩阵各元素之值。

(VI) 从 8480~8496 句, 为求各个系数值, 并计算前后两次结果的差的绝对值和, 若此和小于 0.001, 认为迭代结束, 否则继续。

(VII) 从 8500~8550 句, 计算新的加权阵, 继续转入 8228 句, 进行新的迭代运算。

本程序使用以下数组:

$R(I)$  ——系统的实频特性数据;

$I(I)$  ——系统的虚频特性数据;

$W(I)$  ——实验的各个频率值;

$A(I, J)$  ——正规方程的增广矩阵;

$Q(L)$  ——各个频率的加权值;

$B(I, 1)$  ——上一次求出的各个系数值。

## 三、使用操作

(I) 用 BASIC 引入本程序纸带。本程序的前面部分为产生一个假想系统的实频特性和虚频特性数据存放在数组  $R(I)$  和  $I(I)$  之中。若使用者要输入自己的某个系统的实频特性数据和虚频特性数据, 则要改动语句标号为 8018~8036 的几句。

(II) 用键盘命令 RUN 启动本程序执行, 机器将自动印出: "INPUT N, M", 询问系统的阶数  $N$  和实验数据组数  $M$  (每组有一个频率值, 一个实频数据, 一个虚频数据)。

(III) 使用者用键盘回答  $N$  和  $M$  值之后, 机器又自动印出: "INPUT W (I);", 要求输入实验的各个频率值。当使用者用键盘回答各个频率值后, 机器自动产生假想系统的实频特性数据和虚频特性数据 (否则也要使用者输入给机器)。然后机器进行计算迭代, 每次迭代都印出  $a_i$  和  $b_i$  的中间值, 直到满足精度要求为止。

## 四、试题:

(1) 第一个假想的系统的传递函数为:

$$G(s) = \frac{10 + 4s + 7s^2}{1 + 3s + 5s^2 + 6s^3}$$

∴ 8018~8024 语句为:

8018 DEF FNA (X) = 10 - 7 \* X \* X

8020 DEF FNB (X) = 4 \* X

8022 DEF FNC (X) = 1 - 5 \* X \* X

8024 DEF FND (X) = 3 \* X - 6 \* X \* X \* X

(2) 第二个假想的系统的传递函数为:

$$G(s) = \frac{6 + 7s + 4s^2 + 9s^3}{1 + 7s + 6s^2 + 5s^3 + 10s^4 + 3s^5}$$

∴ 8018~8024 语句改为:

```
8018 DEF FNA (X) = 6 - 4 * X * X
8020 DEF FNB (X) = 7 * X - 9 * X * X * X
8022 DEF FNC (X) = 1 - 6 * X * X + 10 * X ↑ 4
8024 DEF FND (X) = 7 * X - 5 * X ↑ 3 + 3 * X ↑ 5
```

## 五、程序清单及运行结果

```
8000 PRINT "INPUT N, M"                                8000-8014 数据输入
8002 INPUT N, M
8004 PRINT
8006 DIM A(2 * N + 1, 2 * N + 2), W(M + 1), R(M + 1), I(M + 1), B(2 * N + 1, 2)
8008 PRINT "INPUT W (I) "
8010 FOR I=1 TO M
8012     INPUT W(I)
8014 NEXT I
8016 PRINT
8018 DEF FNA (X) = 10 - 7 * X * X                        8018-8036模拟产生一个系统的
8020 DEF FNB (X) = 4 * X                                实频特性数据送R(I); 虚频特
8022 DEF FNC (X) = 1 - 5 * X * X                        性数据送I(I)。
8024 DEF FND (X) = 3 * X - 6 * X * X * X
8026 FOR I=1 TO M
8028     LET B1=W (I)
8030     LET B2=FNC(B1) * FNC(B1)+FND(B1) * FND(B1)
8032     LET R(I)= (FNA(B1) * FNC(B1)+FNB(B1) * FND(B1)) /B2
8034     LET I(I)= (FNB(B1) * FNC(B1)-FNA(B1) * FND(B1)) /B2
8036 NEXT I
8038 GOTO 8220
8040 FOR K=1 TO 2 * N                                    8040-8144列主元消去 法解
8042     LET C=0                                          线性方程组子程序
8044     FOR I=K TO 2 * N
8046         IF ABS (A(I, K)) <ABS(C+.000001) THEN GOTO 8052
8048         LET C=A(I, K)
8050         LET U1=I
8052     NEXT I
8054     IF ABS(C) <.000001 THEN GOTO 8126
8056     LET I=U1
8058     IF I=K THEN GOTO 8070
```

```

8060   FOR J=K TO 2 * N+1
8062       LET T=A(K,J)
8064       LET A(K,J) =A(I,J)
8066       LET A(I,J) =T
8068   NEXT J
8070   LET T=1/A(K,K)
8072   LET A(K,K) =1
8074   FOR J=K+1 TO 2 * N+1
8076       LET A(K,J) =T * A(K,J)
8078   NEXT J
8080   IF K=2 * N THEN GOTO 8092
8082   FOR I=K+1 TO 2 * N
8084       FOR J=K+1 TO 2 * N+1
8086           LET A(I,J) =A(I,J) -A(I,K) * A(K,J)
8088       NEXT J
8090   NEXT I
8092   NEXT K
8094   GOTO 8114
8096   FOR I=1 TO 2 * N
8098       FOR J=1 TO 2 * N+1
8100           IF J<I THEN GOTO 8106
8102           PRINT A(I,J),
8104           GOTO 8108
8106           PRINT T,
8108       NEXT J
8110       PRINT
8112   NEXT I
8114   FOR I=2 * N TO 1 STEP -1
8116       FOR J=I+1 TO 2 * N+1
8118           LET A(I,2 * N+1) =A(I,2 * N+1) -A(I,J) * A(J,2 * N+1)
8120       NEXT J
8122   NEXT I
8124   GOTO 8134
8126   PRINT "NO UNIQUE SOLUTION"
8128   LET T9=T9+1
8130   LET T=1E+6
8132   GOTO 8072
8134   LET C=0
8136   LET T=2 * N-T9
8138   FOR I=1 TO 2 * N

```



```

8140 PRINT A(I, 2 * N + 1);
8142 NEXT I
8144 RETURN
8146 REM CALCULATION  $K_2 = (-1)^{\uparrow K_1}$ 
8148 IF  $K_1 = 0$  THEN GOTO 8162
8150 LET  $K_2 = 1$ 
8152 FOR  $I_1 = 1$  TO  $K_1$ 
8154 LET  $K_2 = (-1) * K_2$ 
8156 NEXT  $I_1$ 
8158 LET  $K_2 = \text{SGN}(K_2)$ 
8160 RETURN
8162 LET  $K_2 = 1$ 
8164 RETURN
8166 REM CALCULATION  $K_3 = \text{SUM}(W(I_1)^{\uparrow K_1})$ 
8168 LET  $K_3 = 0$ 
8170 FOR  $I_1 = 1$  TO M
8172 LET  $K_3 = K_3 + W(I_1)^{\uparrow K_1} * Q(I_1)$ 
8174 NEXT  $I_1$ 
8176 RETURN
8178 REM CALCULATION  $K_3 = \text{SUM}(T(I_1) * W(I_1)^{\uparrow K_1})$ 
8180 LET  $K_3 = 0$ 
8182 FOR  $I_1 = 0$  TO M
8184 LET  $K_3 = K_3 + I(I_1) * W(I_1)^{\uparrow K_1} * Q(I_1)$ 
8186 NEXT  $I_1$ 
8188 RETURN
8190 REM CALCULATION  $K_3 = \text{SUM}(R(I_1) * W(I_1)^{\uparrow K_1})$ 
8192 LET  $K_3 = 0$ 
8194 FOR  $I_1 = 1$  TO M
8196 IF  $K_1 = 0$  THEN GOTO 8202
8198 LET  $K_3 = K_3 + R(I_1) * W(I_1)^{\uparrow K_1} * Q(I_1)$ 
8200 GOTO 8204
8202 LET  $K_3 = K_3 + R(I_1) * Q(I_1)$ 
8204 NEXT  $I_1$ 
8206 RETURN
8208 REM CALCULATION  $K_3 = \text{SUM}(R(I_1)^{\uparrow 2} + I(I_1)^{\uparrow 2} * W(I_1)^{\uparrow K_1})$ 
8210 LET  $K_3 = 0$ 
8212 FOR  $I_1 = 1$  TO M
8214 LET  $K_3 = K_3 + (I(I_1) * I(I_1) + R(I_1) * R(I_1)) * W(I_1)^{\uparrow K_1} * Q(I_1)$ 
8216 NEXT  $I_1$ 
8218 RETURN

```

8146-8218为计算正规方程的  
增广矩阵各元素之值所用  
的几个子程序

```

8220 DIM Q (M+1)
8222 FOR I=1 TO M
8224   LET Q (I) =1
8226 NEXT I
8228 LET K= 0
8230 FOR I=1 TO INT((N-1) /2) * 2+1 STEP 2
8232   LET L= 0
8234   FOR J=1 TO INT ((N-1) /2) * 2+1 STEP 2
8236     LET K1=K
8238     GOSUB 8146
8240     LET K4=K2
8242     LET K1=L
8244     GOSUB 8146
8246     LET K5=K2
8248     LET K1=2 * (K+L)
8250     GOSUB 8166
8252     LET A(I,J) =K4 * K5 * K3
8254     LET K1=2 * (K+L)+1
8256     GOSUB 8178
8258     LET A(I,J+N) =K4 * K5 * K3
8260     LET A(N+I,J) =K4 * K5 * K3
8262     LET K1=2 * (K+L)+2
8264     GOSUB 8208
8266     LET A(I+N, J+N)=K4 * K5 * K3
8268     LET L=L+1
8270   NEXT J
8272   LET K=K+1
8274 NEXT I
8276 LET K= 0
8278 FOR I=2 TO INT(N/2) * 2 STEP 2
8280   LET L= 0
8282   FOR J=2 TO INT(N/2) * 2 STEP 2
8284     LET K1=K
8286     GOSUB 8146
8288     LET K4=K2
8290     LET K1=L
8292     GOSUB 8146
8294     LET K5=K2
8296     LET K1=2 * (K+L) +2
8298     GOSUB 8166

```

8222-8226取初始加权阵为 1

8228-8478计算增广矩阵A  
阵各元素之值

```

8300     LET A(I,J)=K4 * K5 * K3
8302     LET K1=2 * (K+L)+3
8304     GOSUB 8178
8306     LET A(I,J+N)=K4 * K5 * K3
8308     LET A(I+N,J)=K4 * K5 * K3
8310     LET K1=2 * (K+L) +4
8312     GOSUB 8208
8314     LET A(I+N,J+N)=K4 * K5 * K3
8316     LET L=L+1
8318     NEXT J
8320     LET K=K+1
8322     NEXT I
8324     LET K= 0
8326     FOR I=1 TO INT((N-1)/2) * 2+1 STEP 2
8328         LET L= 0
8330         FOR J=2 TO INT(N/2) * 2 STEP 2
8332             LET K1=K
8334             GOSUB 8146
8336             LET K4=K2
8338             LET K1=L
8340             GOSUB 8146
8342             LET K5=K2
8344             LET K1=2 * (K+L)+2
8346             GOSUB 8190
8348             LET A(I,J+N)=K4 * K5 * K3
8350             LET K1=L+1
8352             COSUB 8146
8354             LET K5=K2
8356             LET A(I+N,J)=K4 * K5 * K3
8358             LET L=L+1
8360         NEXT J
8362         LET K=K+1
8364     NEXT I
8366     LET K= 0
8368     FOR I=2 TO INT(N/2) * 2 STEP 2
8370         LET L= 0
8372         FOR J=1 TO INT((N-1)/2) * 2+1 STEP 2
8374             LET K1=K
8376             GOSUB 8146
8378             LET K4=K2

```

```

8380     LET K1=L+1
8382     GOSUB 8146
8384     LET K5=K2
8386     LET K1=2*(K+L)+2
8388     GOSUB 8190
8390     LET A(I,J+N)=K4*K5*K3
8392     LET K1=L
8394     GOSUB 8146
8396     LET K5=K2
8398     LET A(I+N,J)=K4*K5*K3
8400     LET L=L+1
8402     NEXT J
8404     LET K=K+1
8406     NEXT I
8408     LET K=0
8410     FOR I=1 TO INT((N-1)/2)*2+1 STEP 2
8412         LET K1=K
8414         GOSUB 8146
8416         LET K4=K2
8418         LET K1=I-1
8420         GOSUB 8190
8422         LET A(I, 2*N+1)=K4*K3
8424         LET K=K+1
8426     NEXT I
8428     LET K=0
8430     FOR I=2 TO INT(N/2)*2 STEP 2
8432         LET K1=K
8434         GOSUB 8146
8436         LET K4=K2
8438         LET K1=I-1
8440         GOSUB 8178
8442         LET A(I,2*N+1)=K4*K3
8444         LET K1=I
8446         GOSUB 8208
8448         LET A(I+N,2*N+1)=K4*K3
8450         LET K=K+1
8452     NEXT I
8454     GOTO 8480
8456     FOR I=1 TO 2*N
8458         FOR J=1 TO 2*N+1

```

```

8460     PRINT A(I,J) ,
8462     NEXT J
8464     PRINT
8466     NEXT I
8468     PRINT
8470     FOR I=1 TO 2 * N
8472     FOR J=1 TO 2 * N+1
8474         LET B(I,J) =A(I,J)
8476     NEXT J
8478     NEXT I
8480     LET T9=0
8482     GOSUB 8040
8484     PRINT
8486     LET Q1=0
8488     FOR I=1 TO 2 * N
8490         LET Q1=Q1+ABS(A(I,2 * N+1)-B(I,1))
8492         LET B(I,1) =A(I,2 * N+1)
8494     NEXT I
8496     IF Q1<.001 THEN GOTO 8552
8498     FOR L=1 TO M
8500         LET K=1
8502         LET Q1=1
8504         FOR I=2 TO INT(N/2) * 2 STEP 2
8506             LET K1=K
8508             GOSUB 8146
8510             LET K4=K2
8512             LET Q1=Q1+K4 * W(L) ↑ I * A(N+I,2 * N+1)
8514             LET K=K+1
8516         NEXT I
8518         LET K=0
8520         LET Q2=0
8522         FOR I=1 TO INT(N/2) * 2+1 STEP 2
8524             LET K1=K
8526             GOSUB 8146
8528             LET K4=K2
8530             LET Q2=Q2+K4 * W(L) ↑ I * A(N+I, 2 * N+1)
8532             LET K=K+1
8534         NEXT I
8536         LET Q(L)=1/(Q1 * Q1+Q2 * Q2)
8538     NEXT L

```

解线性方程组

8486—8496 求前后两次得到的各个系数值的差的绝对值和  $Q_1$ , 若  $Q_1 < 0.001$  结束 否则继续。

8498—8550 计算新的加权阵之值

```

8540 FOR I=1 TO 2*N
8542   FOR J= 1 TO 2*N+1
8544     LET A(I,J)= 0
8546   NEXT J
8548 NEXT I
8550 GOTO 8228
8552 END

```

运行结果

(1) 第一试题运行结果

```

RUN
INPUT N, M
? 3? 20
INPUT W (I) :
? .001? .005? .01? .015? .02? .04? .09? .1? .5? 1? 2? 4? 6
? 9? 10? 14? 16? 20? 25? 29
SOLUTION:
9.94043 3.9963 6.87888 2.98426 4.96995 5.89606
SOLUTION:
10 3.99989 7.00015 3 4.99997 6.00003
SOLUTION:
9.99999 3.99985 6.99996 2.99995 4.99997 5.99979
END AT 8552

```

(2) 第二试题运行结果

```

RUN
INPUT N, M
? 5? 20
INPUT W (I) :
? .001? .005? .01? .015? .03? .05? .09? .1? .5? 1? 2? 3? 5
? 9? 10? 15? 20? 25? 30? 35
SOLUTION:
5.56081 -.310469 .346259 5.29269E-4 -4.54096E-7 5.84124
1.5758 .230719 .116165 1.78894E-4
SOLUTION:
6.00444 10.2659 3.08607 12.4942 -5.71907E-4 7.5739 8.67993
4.65979 13.0827 4.16153
SOLUTION:
6.00001 7.03581 3.98848 9.03751 -7.25746E-4 7.00601 6.03134
4.98955 10.0356 3.01074
SOLUTION:

```

5.99997 6.95368 4.01447 8.95184 6.09636E-4 6.99222 5.95965  
5.0132 9.95524 2.98594

SOLUTION,

6.00001 7.02768 3.99128 9.02871 -3.78132E-4 7.00465 6.02411  
4.99205 10.0267 3.00838

SOLUTION,

6 7.0145 3.99542 9.01497 -1.90973E-4 7.00244 6.01263  
4.99578 10.0138 3.00437

END AT 8552

## 六、注意事项

本程序在频率跨几个倍程时, 计算结果还是相当精确, 对已知系统的阶次以后, 本程序十分好用。当系统的阶次N不知道时, 本程序计算结果偏差较大。

## 本章参考文献

[1] 绪方胜彦著

《现代控制工程》

科学出版社1976

[2] C.K.Sanathan and J.koerner

《Transfer Function synthesis as a

Ratio of Two Complex polynomials》

IEEE Trans. Automatic Control

Vol.AC-8, pp.56-58, Jannary 1963

[3] P.A.Payne

《An Improved Technique for Transfer Function

synthesis From Frequency Response Data》

IEEE Trans Automatic Control pp.483-482, August 1970

## 第十九章 随机服务系统的仿真与马尔可夫链分析

### §1. 随机服务系统简介

就系统而言,可以粗略地划分为确定性的和随机性的两大类。确定性系统,它在每一时刻的状态变化是可预知的;而随机性系统,它在每个时刻的状态变化是预先不可知的。运筹学中所涉及的系统大都属于随机性的,又称为随机服务系统。比如一个工厂可以看作一个随机服务系统,它服务的对象是用户,而原材料的采购、制造加工、装配、销售等为服务过程。用户定货是随机的。对这样系统我们可以用仿真研究在现有的用户定货情况下,生产应如何组织、调度,以及工厂规模是否要扩大或缩小,才使经济收益最大。又如一个电信局的电话系统,也是一个随机服务系统。此系统服务的对象是打电话的人,电话转接、使用可以看成服务过程。人们使用电话是随机的。对这样的系统进行仿真研究可以确定某地区要新建一个电信局,这个电信局要配多少条电话线,多少条转接线,多少部电话,才能满足本地区人们的需要并且又最经济。或者确定某个电信局现有电话线路使用率有多高,顾客排队等待的时间有多长以及经济指标有多高等等。

总之,随机服务系统遍及各行各业。要合理地、有效地、经济地生产、管理、销售,对各种各样的随机服务系统加以认真研究,是十分必要的。仿真确定性系统,比较有效的数学方法是尤格库塔法;而仿真随机服务系统,比较有效的方法是蒙特卡洛法(Monte Carlo Methods)。

### §2 单队单服务员的随机服务系统仿真

#### 一、计算方法概述

一个打字员打印文件,就是一个单队单服务员的随机服务系统。服务对象是要打印文件的人,而其收件、打字等是服务过程,文件送来是随机的。仿真研究这个系统,主要了解当前这个打字员忙闲情况以及一个文件要等多长时间才能打完,从而决定安排几个打字员为好。

和确定性系统的仿真工作一样,仿真的第一步就是构造仿真模型。系统仿真是否真实,主要在于模型构造是否合理。要建立数学模型就要首先收集数据。就打字员打印文件这个系统来说,根据实际情况,收集数据如下表:(见表19-1)

在本例中,所谓建立数学模型就是构造文件连续到达的时间间隔分布和对每个文件服务时间的分布。在此基础上假设文件来自无限源并排成单队,在先来先服务的原则之下不限制服务时间长短,从而产生仿真过程中需要的“顾客到达时间数”和“服务时间数”两个随机变量的抽样值。

根据收集到的数据,至少有三种方法产生仿真所需要的随机变量的抽样值。

(1) 使用实际数据。为了使仿真结果反映系统在任何条件下的真实属性,要求收集的数据充分多才行。所谓充分多,就是保证数据中各种可能的条件都要包括到。显然用这种方



表 19-1

文件排队等待打字的有关数据  
(8:00开始记录)

文件编码	收到时间	开始打字时间	完成打字时间	与收到前一个文件的间隔时间	服务时间
1	8:07	8:07	8:17	7	10
2	8:38	8:38	9:07	31	29
3	8:39	9:07	9:27	1	20
4	8:44	9:27	9:44	5	17
5	9:22	9:44	9:53	38	9
6	9:25	9:53	10:09	3	16
7	9:28	10:09	10:26	3	17
8	9:49	10:26	10:30	21	4
9	10:06	10:30	10:39	17	9
10	10:22	10:39	10:47	16	8
11	10:25	10:47	10:55	3	8
12	11:03	11:03	11:23	38	20
13	12:04	12:24	12:42	1	18
14	12:29	12:42	12:52	25	10
15	12:43	12:52	1:16	14	24
16	12:44	1:16	1:21	1	5
17	1:04	1:21	1:32	20	11
18	1:16	1:32	1:44	12	12
19	3:06	3:06	3:21	94	15
20	3:50	3:50	4:37	44	47

法构造一个模型太费人力太费时间了。

(2) 由原始收集的数据, 画出统计量的累积分布的直方图, 然后利用作出的分布图产生统计量的抽样值。就本例来说, 通过记录的20个文件的实际记录的数据, 分别作出文件连续到达的间隔时间累积分布图(图19-2-1)和服务时间累积分布图(图19-2-2)

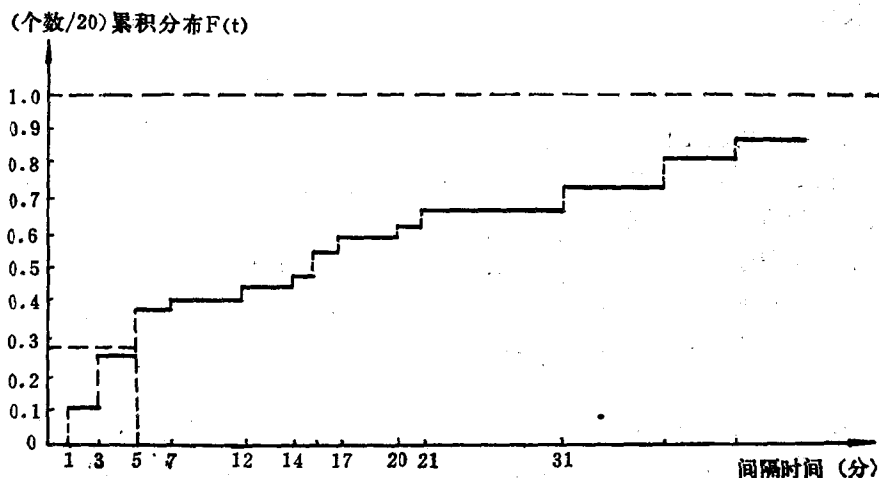


图 19-2-1 文件连续到达的间隔时间累积分布图

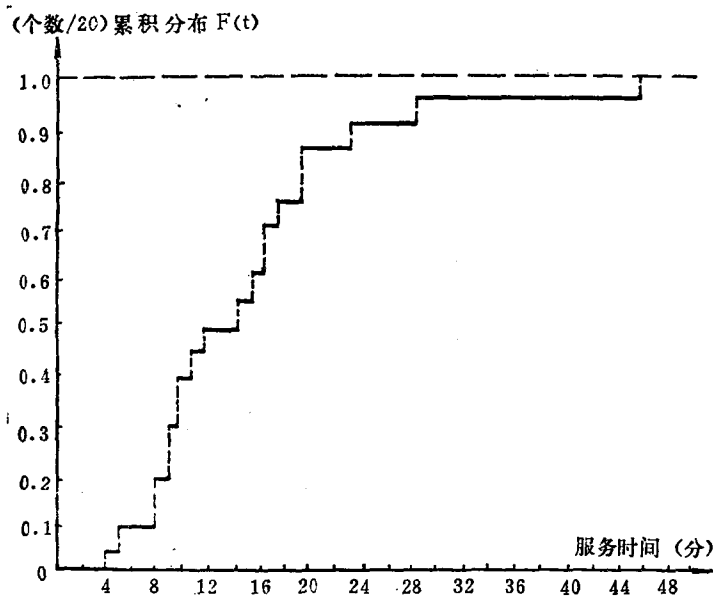


图 19-2-2 服务时间累积分布图

图19-2-1是这样做出来的, 即从原始记录数据中找出隔1分钟来的文件有2个,  $2/20=0.1$ 作为第一点; 隔3分钟来的文件有3个, 则 $0.1+3/20=0.25$ 作为第二点; 隔5分钟来的文件有2个, 则 $0.25+2/20=0.35$ 作为第三点……。

图19-2-2用同样的方法, 画出服务时间的累积分布图。

有了这两个图, 如何产生仿真需要的随机变量抽样值呢? 抽样的方法是这样的, 使用某种方法产生0—1之间的均匀分布的随机数 (在 BASIC 语言中有标准函数 RND(X) 可以使用)。每产生一个均匀分布的随机数, 就分别标在图19-2-1和图19-2-2的  $F(t)$  轴上, 然后作水平线交在累积分布曲线上或其不连续段上, 从而由时间轴上读出相应的  $t$  值。举例来说, 若产生的随机数为0.27, 在图19-2-1的  $F(t)$  轴上找到0.27, 从该点向右画水平线, 与累积分布曲线不连续段相交, 得到相应的时间  $t=5$ , 就被选作为“与前一个文件到达间隔时间”。相应的从图19-2-2中产生服务时间。如此进行下去, 任选一个0—1之间的随机数, 就可以得到一个到达间隔时间和一个服务时间的抽样值。

(3) 先假设实际数据服从一定的理论分布, 然后可以从假设的理论分布中进行抽样, 来得到系统仿真所需要的输入数据。一般来讲, 在多数情况下这种方法比前面两种要好。那么怎样确定与实际数据较近似的理论分布呢? 可以先考虑几种可能的分布作为候选, 然后用  $\chi^2$  检验, 以决定选用其中“最好”的一种理论分布。一般在多数情况下, 顾客到达都是随机的, 所以单位时间到达的个数是满足某种泊松分布 (Poisson)。这就意味着接踵而来的两个文件到达的间隔时间大小是服从具有和泊松分布相同参数的指数分布。因此间隔时间的均值就可以用泊松分布这个参数来近似。不过为确定到达时间间隔的分布还要进行检验 (限于篇幅, 有关检验的方法和理论不在此多述)。就打字员打印文件这个例子来看, 我们将假设文件到达的时间间隔为指数分布。由实验数据可以计算出文件到达的时间间隔的平均值为:  $\bar{t}_a=22.7$  分钟。所以到达时间间隔的分布假设为如下形式的指数分布,

$$f(t_a) = \frac{1}{22.7} e^{-\sqrt{(t_a/22.7)}}$$

$$=0.044053 e^{-0.044053 t_s} \quad t_s > 0 \quad (19-2-1)$$

$$E(t_s) = 22.7 \text{ 分钟}$$

式 (19-2-1) 即为间隔时间的分布的解析式, 即所谓系统仿真的数学模型之一。

那么服务时间分布又怎么确定呢? 一般来说, 某种爱尔朗分布 (Erlang) 将很好地近似实际的服务时间分布, 通过检验我们可以确定爱尔朗分布的最好  $k$  值。在本例中我们不妨假设  $k=1$  的爱尔朗分布较好地近似实际的服务时间分布。这样得到的是又一个指数分布。由实验数据可以计算出实际的服务时间的均值为:  $\bar{t}_s = 15.45$  分钟, 这个值就近似地等于服务时间的指数分布的参数。于是, 得到服务时间分布的指数分布形式为:

$$\begin{aligned} g(t_s) &= \frac{1}{15.45} e^{-\frac{t_s}{15.45}} \\ &= 0.064725 e^{-0.064725 t_s} \quad t_s > 0 \end{aligned} \quad (19-2-2)$$

$$E(t_s) = 15.45 \text{ 分钟}$$

式 (19-2-2) 即为服务时间分布的解析式, 即所谓系统仿真的数学模型之二。

有了式 (19-2-1) 和式 (19-2-2), 我们就可以应用蒙特卡洛法, 找出  $T_s$  和  $T_s$  的抽样值。

由式 (19-2-1) 可推导出

$$\begin{aligned} F(t_s) &= p(x < t_s) = \int_0^{t_s} 0.044053 e^{-0.044053 x} dx \\ &= -e^{-0.044053 x} \Big|_0^{t_s} \\ &= 1 - e^{-0.044053 t_s} \\ \text{令 } r &= F(t_s) = 1 - e^{-0.044053 t_s} \\ e^{-0.044053 t_s} &= 1 - r \\ -0.044053 t_s &= \ln(1 - r) \\ \therefore t_s &= -\frac{1}{0.044053} \ln(1 - r) \end{aligned} \quad (19-2-3)$$

其中  $r$  为  $0 \sim 1$  之间均匀分布的随机数。

当  $r = 0.27$  时, 可以从上式得出到达时间间隔为 7 分钟, 这和前面根据直方图 (图 19-2-1) 得到的 5 分钟是相近的。由于本例中收集的数据比最起码的要求还要少得多, 因此我们不能指望用这些数据就能代表真实系统, 在此, 只不过用以说明仿真进行的步骤。要想能很好地代表真实系统, 数据收集工作至少要进行 2、3 个星期。

和时间间隔抽样类似, 服务时间抽样也可以导出如下公式:

$$t_s = -15.45 \ln(1 - r) \quad (19-2-4)$$

由于  $r$  是  $0 \sim 1$  之间均匀分布的随机数,  $r$  和  $1 - r$  是等可能的, 所以, 用以下两个公式产生到达时间间隔和服务时间完全可以代替 (19-2-3) 和 (19-2-4) 两式。

$$t_s = -22.7 \ln(r) \quad (19-2-5)$$

$$t_s = -15.45 \ln(r) \quad (19-2-6)$$

在这里需说明一点, 求  $t_i$  用的  $r$  值和求  $t_{i+1}$  用的  $r$  值不能是同一个值。

一旦建立了仿真数学模型, 下一步就是选择仿真的方法。基本上常用的有两种方法, 这两种方法的区别在于仿真过程中仿真时间增加的方式不同, 即:

(1) 在仿真过程中, 仿真时间是按单位时间增加的(如分、秒等)。在仿真过程中, 每增加一个单位时间, 就要询问系统中有随机事件发生没有, 如果没有, 仿真时间再增加一个单位时间; 如果有随机事件发生, 系统的状态就发生变化。

(2) 在仿真过程中, 仿真时间是按“发生事件的时刻”跳跃式地向上增加的。在仿真一开始, 首先判断那个事件最早发生, 仿真时间立即从初始时刻变成下一个事件发生的时刻, 这样仿真时间总是从某一个事件发生的时刻跃增到下一个事件发生的时刻。

我们的打字员打印文件的例子中, 只有两个事件发生, 因此采用了按事件发生的时刻增加仿真时间的方法进行仿真的。

## 二、程序说明:

编制此程序时作了如下的假设: 在  $t = 0$  时第一个文件已经到达; 先到者先服务的排队规则; 无限的顾客流; 等待时间不受限制。

编制出的程序框图如图19-2-3。

程序中使用的变量说明如下:

$A_1$ ——下一个顾客到达时刻。

$A_2$ ——顾客到达的指数分布的均值。

$A_3$ ——服务时间的指数分布的均值。

$S$ ——下一个顾客离开服务的时刻。

$C$ ——当前时刻。(从仿真开始时刻算起)

$M_1$ ——进行仿真的最大时间。

$N_1$ ——队中现有顾客数。

$T_1$ ——服务员空闲的总时间。

$N_2$ ——进入队伍的顾客总数。

$T_2$ ——所有顾客总的等待时间。

$T_3$ ——所有顾客在服务台停留的时间。

$T_4$ ——所有顾客在系统中的总时间。

$N_3$ ——顾客总数。

$M_2$ ——最长的队长。

$A_4$ ——顾客在服务台停留的平均时间。

$A_5$ ——顾客排队平均时间。

$A_6$ ——排队等待的顾客在队中的平均时间。

$A_7$ ——顾客在排队中停留的平均时间。

$P$ ——服务员忙着的时间的百分数。

## 三、使用方法及上机操作:

(1) 本程序只适用于仿真单队单服务员的随机服务系统仿真。若使用者仿真的随机服务系统中随机事件的概型与本例不一样, 要先修改程序中的第八、九两句(语句标号为8014和8016), 程序中其他语句可以不必改动就可以使用。

(2) 用 BASIC 程序将本程序输入机器内, 然后按(1)中所述修改源程序。程序修

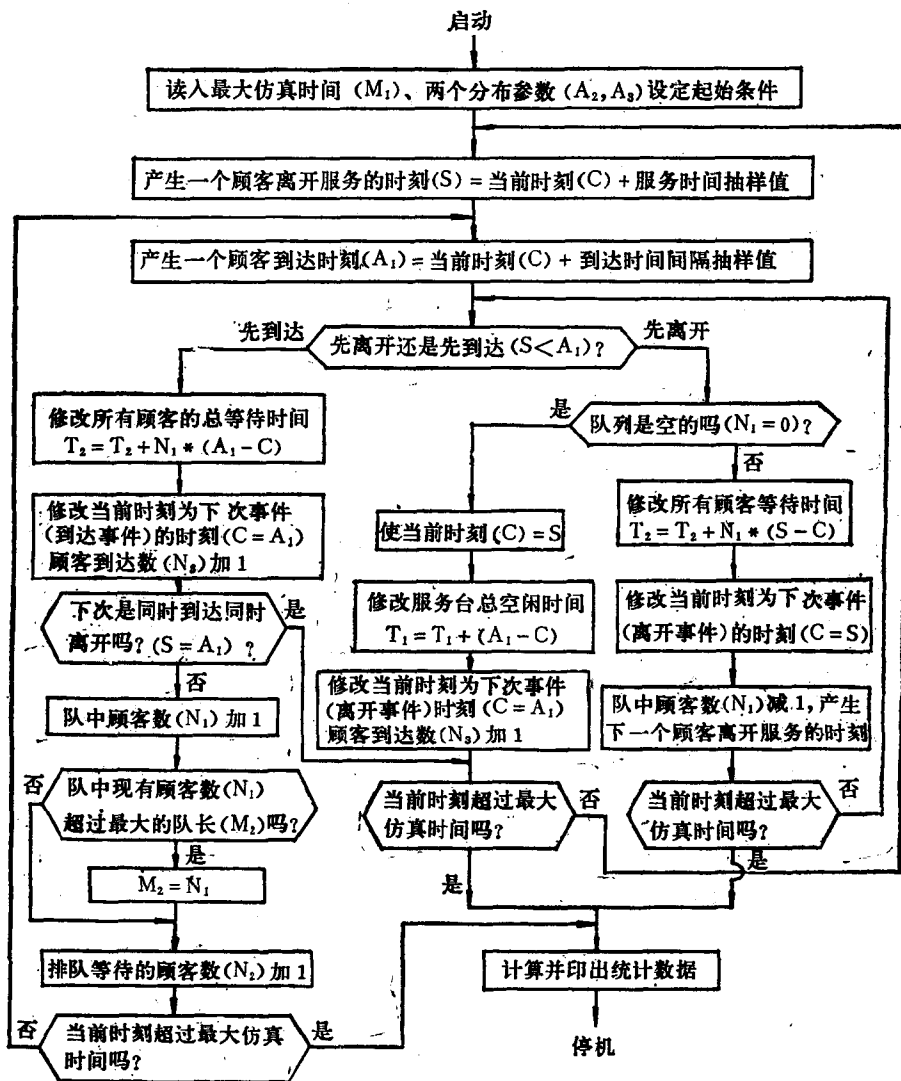


图 19-2-3 单队单服务员随机服务系统仿真程序框图

改好后用键盘命令 RUN 启动本程序，机器将自己印出：“START,” 表示仿真开始。然后又自动印出：“INPUT A<sub>2</sub>, A<sub>3</sub>, M<sub>1</sub>”，分别询问两个随机变量的指数分布参数以及最大仿真时间。

(3) 用键盘回答以上各值后，机器自动进行仿真计算，待运行到最大仿真时间后印出仿真结果信息如下，按印出顺序从左到右分别是

- T<sub>3</sub>——所有顾客在服务台停留的时间总和；
- T<sub>4</sub>——所有顾客在系统中的总时间；
- A<sub>7</sub>——顾客在系统中停留的平均时间；
- A<sub>4</sub>——顾客在服务台停留的平均时间；
- A<sub>5</sub>——顾客排队的平均时间；
- A<sub>6</sub>——排队等待的顾客们在队中的平均时间；
- T<sub>1</sub>——服务员空闲的总时间；

P——服务台忙着的时间的百分数;

$M_2$ ——最大的队长。

后面附有三次运行结果记录。三次仿真时间分别为50000(分钟), 25000(分钟), 100000(分钟)。从结果来看, 打字员还是相当忙碌的, 忙着的时间的百分数在68~70%之间; 而文件等待时间也不短, 若要求文件打印速度快一点的话, 有必要再加一个打字员, 以缩小文件排队等待的时间。

#### 四、程序清单及运行结果

```
8000 PRINT "START:"
8002 PRINT "INPUT A2, A3, M1"
8004 INPUT A2, A3, M1
8006 PRINT
8008 PRINT "ANS,"
8010 PRINT
8012 LET N3=1
8014 LET S=C-(1/A3)*LOG(RND(1))
8016 LET A1=C-(1/A2)*LOG(RND(2))
8018 IF S<A1 THEN GOTO 8046
8020 LET T2=T2+N1*(A1-C)
8022 LET C=A1
8024 LET N3=N3+1
8026 IF S=A1 THEN GOTO 8038
8028 LET N1=N1+1
8030 IF N1>M2 THEN GOTO 8042
8032 LET N2=N2+1
8034 IF M1>C THEN GOTO 8016
8036 GOTO 8070
8038 IF M1>C THEN GOTO 8014
8040 GOTO 8070
8042 LET M2=N1
8044 GOTO 8032
8046 IF N1><0 THEN GOTO 8060
8048 LET C=S
8050 LET T1=T1+A1-C
8052 LET C=A1
8054 LET N3=N3+1
8056 IF M1>C THEN GOTO 8014
8058 GOTO 8070
8060 LET T2=T2+N1*(S-C)
8062 LET C=S
```

```

8064 LET N1=N1-1
8066 LET S=C-(1/A3)*LOG(RND(3))
8068 IF M1>C THEN GOTO 8018
8070 LET T3=C-T1
8072 LET T4=T3+T2
8074 LET A7=T4/N3
8076 LET A4=T3/N3
8078 LET A5=T2/N3
8080 LET A6=T2/N2
8082 LET P=T3/C
8084 PRINT T3, T4, A7, A4, A5, A6, T1, P, M2
8086 END

```

第一次运行结果:

RUN

START,

INPUT A2, A3, M1

? .044053 ? .064725 ? 25000

ANS,

17552.8	59912.4	53.7813	15.7566
38.0248	54.4468	7451.02	.702005

16

END AT 8086

第二次运行结果:

RUN

START,

INPUT A2, A3, M1

? .044053 ? .064725 ? 50000

ANS,

34315.4	101446	45.7377	15.4713
30.2663	44.1069	15761.1	.68526

12

END AT 8086

第三次运行结果:

RUN

START,

INPUT A2, A3, M1

? .044053 ? .064725 ? 100000

ANS,

69085	235748	53.3126	15.623
37.6895	54.8956	30919.6	.690818

• 344 •

### §3 多队单服务员随机服务系统仿真

#### 一、计算方法概述

我们较详细地来看看电信局的电话系统，若一个电话转接员负责 $M$ 条连接线共 $N$ 个电话机，即如下图所示

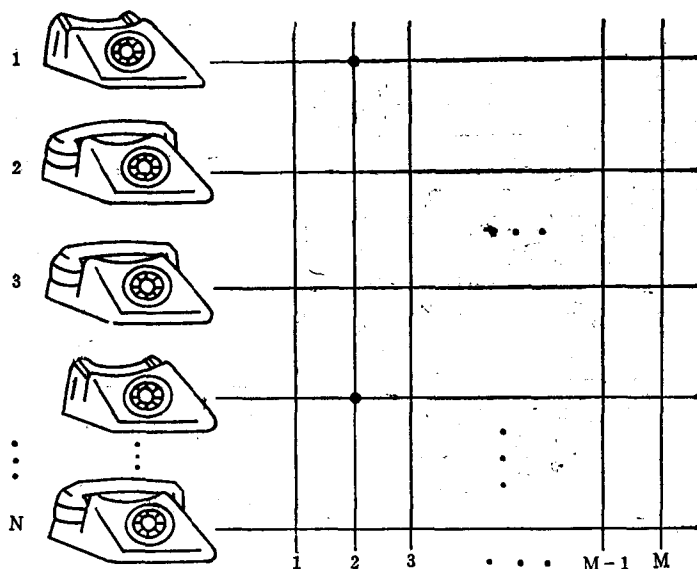


图 19-3-1 电话系统示意图

这样的电话系统就可以看成有 $M$ 队的单个服务员的随机服务系统。

如在§2节中所述，仿真的第一步建立数学模型。在本例中假设每条连接线的电话呼叫都是等概率的，由实际的统计数据确定电话随机呼叫的间隔时间是服从某种泊松分布的，而平均间隔时间统计结果是12秒。同样假设每个电话机通话的时间也是等概率的，由实际的统计数据确定是服从指数分布的，而平均通话时间是120秒。仿真的过程大略如下进行。当有一个电话随机呼叫产生，若呼叫的电话空着，且至少有一根连接线也空着情况下，电话接通这次呼叫是成功的，经过一定通话时间，通话结束，两个电话和连接线又空下来；若电话随机呼叫到来时，或呼叫的电话正在使用，或没有空闲连接线，则这次呼叫失败。通过仿真我们要研究该系统在一定时间内，呼叫多少次，其中成功多少次，失败多少次，其中因终点电话占用失败多少次，因连接线不空失败多少次，以及呼叫成功的百分数等各个统计值。

由于问题的描述、理论推导和§2中所讲的完全相同，因此从略。本例采用的第一种仿真方法，即仿真时间是一秒一秒增加的。

#### 二、程序说明

编制此程序时同样假设：在 $t=0$ 时，没有一个电话在使用；先到先用的排队规则；无限的顾客流；<sup>2</sup>等待时间不受限制等。

编制出的程序框图如图19-3-2



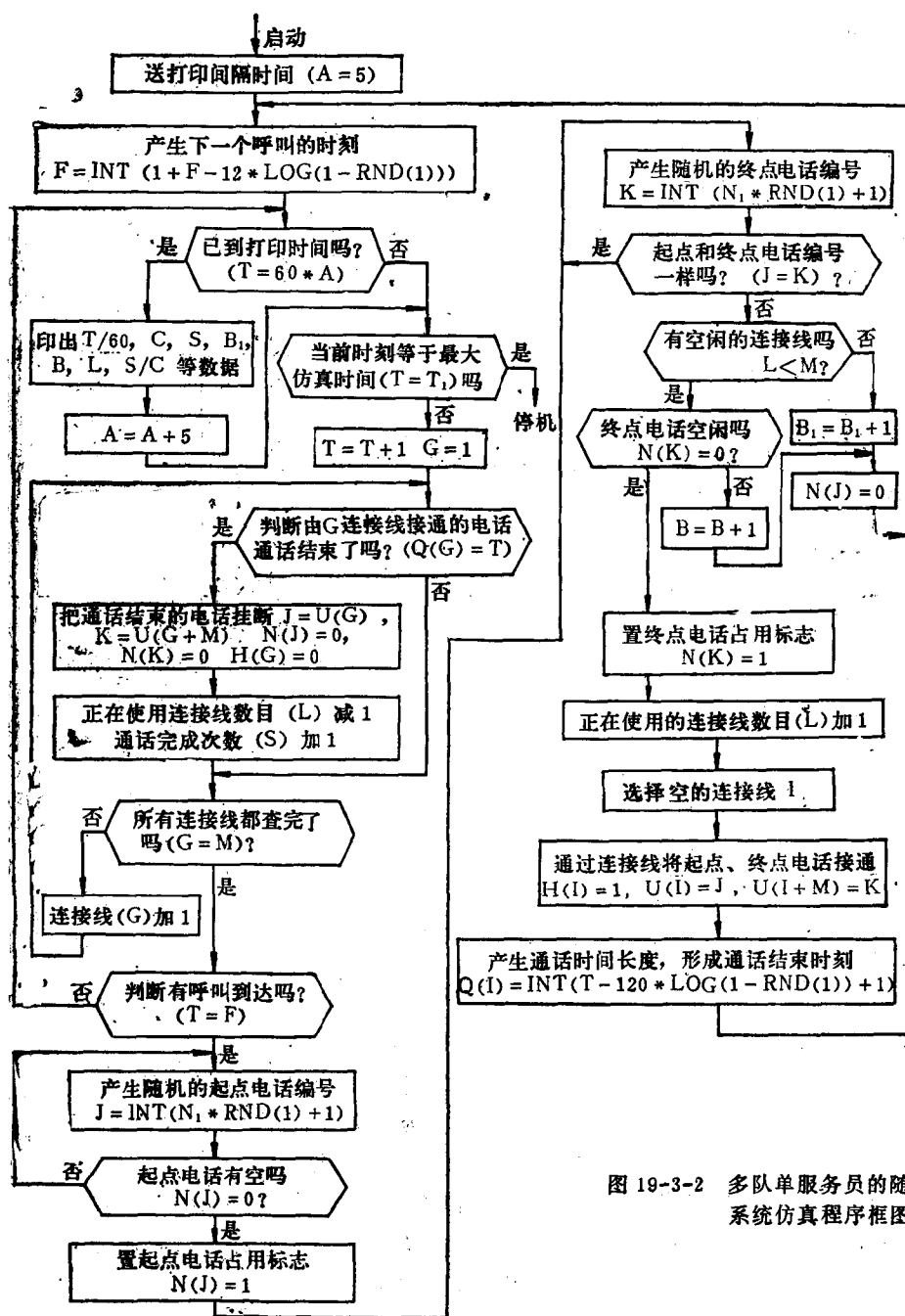


图 19-3-2 多队单服务员的随机服务  
系统仿真程序框图

在程序中使用的变量和数组说明如下:

- F——下一个电话呼叫到达时刻。
- $T_1$ ——系统仿真最大时间。
- T——当前时刻。
- M——电话连接线数目。
- $N_1$ ——电话总台数。

G——连接线的序号。

Q(G)——第G号连接线上通话结束时刻。

J——起点的电话序号。

K——终点的电话序号。

N(I)——第I台电话的状态 ( $N(I)=1$ , 表示正在通话,  $N(I)=0$ , 表示电话闲着)。

H(I)——第I根连接线的状态 ( $H(I)=1$ , 表示正在使用,  $H(I)=0$ , 表示闲着)。

L——正在使用的连接线数目

C——呼叫次数

S——通话完成的次数

B——因终点电话不空而使呼叫失败的次数。

B<sub>1</sub>——因连接线不空而使呼叫失败的次数。

U(I)——第I根连接线所接通的起点电话序号。

U(I+M)——第I根连接线所接通的终点电话序号。

### 三、使用方法及上机操作

1. 若使用者仿真的系统概型不同本例, 要改动8044、8088、8096和8132等句, 其他语句不用修改就可以使用。

2. 用BASIC解释程序引入本程序后, 按(1)修改好源程序, 再用键盘命令RUN启动本程序运行。本程序一旦运行, 相继印出“电话系统仿真”的英文字, 并询问电话机台数N<sub>1</sub>; 连接线数M以及总的仿真时间T<sub>1</sub>。

3. 当使用者用键盘回答以上各值后, 机器又自动印出“START;”以及运行结果信息表的表头各栏的英文字。按从左到右分别是: 时间(分钟), 呼叫次数, 成功次数, 因连接线不空而失败次数, 因终点电话不空而失败次数, 正在使用的连接线数目和成功比例数。往下每仿真过5分钟就把相应仿真结果印在各栏之下, 直到到达总的仿真时间为止。

### 四、程序清单及运行记录。

#### 程序清单

```
8000 REM SIMULATION OF A TELEPHONE SYSTEM
8002 PRINT TAB(20); "....."
8004 PRINT TAB(20); "THE SIMULATION OF A TELEPHONE SYSTEM"
8006 PRINT
8008 PRINT
8010 PRINT TAB(20); "THE MEMBER OF TELEPHONES(N1)="
8012 INPUT N1
8014 PRINT
8016 PRINT TAB(20); "THE NUMBER OF LINKS(M)="
8018 INPUT M
8020 PRINT
8022 PRINT TAB(20); "THE TOTAL SIMULATION TIME(T1)="
8024 INPUT T1
8026 PRINT
8028 PRINT
```

```

8030 PRINT TAB(10); "START; "
8032 PRINT
8034 PRINT TAB(1); "TIME(MIN)"; TAB(13); "CALL; TAB(19); "SUCCESS";
8036 PRINT TAB(29); "BLOCK"; TAB(37); "BUSY"; TAB(45); "LINK";
8038 PRINT TAB(50); "RATIO(S/C)"
8040 DIM Q(M), N(N1), H(M), U(2 * M)
8042 LET A=5
8044 LET F=INT(1+F-12 * LOG(1-RND(1)))
8046 IF T><60 * A THEN GOTO 8054
8048 PRINT TAB(3); T/60; TAB(13); C; TAB(21); S; TAB(30); B1;
8050 PRINT TAB(38); B; TAB(44); L; TAB(50); 100 * S/C
8052 LET A=A+5
8054 IF T=T1 THEN GOTO 8136
8056 LET T=T+1
8058 LET G=1
8060 IF Q(G)><T THEN GOTO 8076
8062 LET J=U(G)
8064 LET K=U(G+M)
8066 LET N(J)=0
8068 LET N(K)=0
8070 LET H(G)=0
8072 LET L=L-1
8074 LET S=S+1
8076 IF G=M THEN GOTO 8082
8078 LET G=G+1
8080 GOTO 8060
8082 IF T=F THEN GOTO 8086
8084 GOTO 8046
8086 LET C=C+1
8088 LET J=INT(N1 * RND(1)+1)
8090 IF N(J)=0 THEN GOTO 8094
8092 GOTO 8088
8094 LET N(J)=1
8096 LET K=INT(N1 * RND(1)+1)
8098 IF J=K THEN GOTO 8096
8100 IF L<M THEN GOTO 8106
8102 LET B1=B1+1
8104 GOTO 8110
8106 IF N(K)=0 THEN GOTO 8114
8108 LET B=B+1

```

```

8110 LET N(J)=0
8112 GOTO 8044
8114 LET N(K)=1
8116 LET L=L+1
8118 LET I=1
8120 IF H(I)=0 THEN GOTO 8126
8122 LET I=I+1
8124 GOTO 8120
8126 LET H(I)=1
8128 LET U(I)=J
8130 LET U(I+M)=K
8132 LET Q(I)=INT(T-120*LOG(1-RND(1))+1)
8134 GOTO 8044
8136 END

```

运行结果

.....  
 THE SIMULATION OF A TELEPHONE SYSTEM  
 THE NUMBER OF TELEPHONES(N<sub>1</sub>)=

↑ 50

THE NUMBER OF LINKS(M)=

↑ 10

THE TOTAL SIMULATION TIME(T<sub>1</sub>)=

↑ 12000

START:

TIME(MIN)	CALL	SUCCESS	BLOCK	BUSY	LINK	RATIO(S/C)
5	27	14	0	3	10	51.8518
10	47	29	2	8	8	61.7021
15	68	47	2	11	8	69.1176
20	83	62	2	15	4	74.6988
25	105	79	2	16	8	75.2381
30	133	100	2	22	9	75.188
35	157	122	2	28	5	77.707
40	181	138	2	36	5	76.2431
45	201	155	3	37	6	77.1144
50	229	167	9	43	10	72.9258
55	254	190	11	48	5	74.8031
60	277	204	13	52	8	73.6462
65	299	219	20	54	6	73.2441
70	322	236	20	60	6	73.2919

75	347	258	20	64	5	74.3516
80	377	276	23	69	9	73.2095
85	408	298	25	78	7	73.092
90	436	316	26	86	8	72.4771
95	461	331	28	92	10	71.8004
100	491	350	33	101	7	71.2831
105	522	371	33	111	7	71.0728
110	537	385	33	114	5	71.6946
115	558	403	33	120	2	72.2222
120	585	421	33	126	5	71.9658
125	613	438	33	135	7	71.4519
130	633	451	35	137	10	71.248
135	657	466	38	145	8	70.9285
140	683	484	40	152	7	70.8638
145	707	500	40	159	8	70.7213
150	737	519	41	168	9	70.4206
155	760	535	43	174	8	70.3947
160	785	554	45	182	4	70.5732
165	809	568	46	186	9	70.2101
170	836	587	48	192	9	70.2153
175	854	604	51	197	2	70.726
180	884	619	51	206	8	70.0226
185	915	634	51	222	8	69.2896
190	945	653	58	230	4	69.1005
195	964	664	58	235	7	68.8797
200	981	681	58	240	2	69.4189

END AT 8136

#### 参考文献:

- [1] Billy.E.Gillett  
《Introduction to Operations Research, A Computer-oriented Algorithmic Approach》1976
- [2] 徐光辉《随机服务系统》1980
- [3] 孙国基等  
《离散系统的计算机仿真》  
西安交大科学技术报告79—223 1979
- [4] 张建中  
《蒙特卡洛方法》  
数学的实践与认识 1974. 1—2期

## §4 马尔可夫链分析

### 一、计算方法概要

在实际的系统研究中,有一类系统基本上可以使用确定的数学模型来描述,只是增加一些随机的扰动而已;而另外一类系统,基本上就要使用随机性的模型来描述。研究这些随机性的问题就是研究“随机过程”。

随机过程分为连续的随机过程和离散的随机过程。所谓连续的随机过程可以用 $\{X(t), t \in T\}$ 来表示,其中 $X(t)$ 是对每个时刻的随机过程的取值,称为状态;而 $T$ 表示时间 $t$ 的变化范围,可以是有限值,也可以是无限制的。(参数 $t$ 可以指时间,也可以指空间或者与随机现象有关的可测参数)而离散的随机过程可以用 $\{X(n), n=0, 1, 2, \dots\}$ 表示,这里 $n$ 为参数, $X(n)$ 是状态。下面我们主要讨论离散随机过程。

最简单的离散随机过程就是二项过程。这个过程是这样的。例如对一批产品进行检验分类,若检查第 $n$ 个产品为正品,则用 $X(n)=0$ 表示;若第 $n$ 个产品为次品,则用 $X(n)=1$ 表示。又知出现次品的概率为 $P(X(n)=1)=p$ ,则出现正品的概率 $P(X(n)=0)=1-p=q$ ,则 $n$ 次检验中 $x$ 次出现次品的概率为:

$$\binom{n}{x} p^x q^{n-x}, \text{ 其中 } \binom{n}{x} = \frac{n!}{x!(n-x)!}.$$

这种过程就叫做二项过程。在二项过程中状态变量之间是相互独立的,因此是最简单的离散随机过程。二项过程可以用贝努里(Bernoulli)概型来描述。

可是大多数离散随机过程中,各状态之间并不是相互独立的,因此就使问题复杂化了。在较复杂的随机过程中,人们研究的最多、应用又较广的就是马尔可夫过程,它首先由 A.A. Markov 在 1906 年研究的,因此称为马尔可夫模型。

马尔可夫过程是由马尔可夫性质确定的。什么叫做马尔可夫性质呢?就是随机过程的“无后效性”,关于无后效性我们可以作如下的说明。

由动态规划的最优原理告诉我们,当研究过程从 $X(t_0)$ 状态经过 $X(t_1)$ 状态转移到 $X(t_f)$ 状态,如果从 $X(t_0)$ 状态到 $X(t_f)$ 状态是最优的轨线,则从 $X(t_1)$ 到 $X(t_f)$ 必然也是最优轨线。亦即,不管 $t=t_1$ 时的状态 $X(t_1)$ 是如何达到的,一旦已知 $X(t_1)$ ,则以 $X(t_1)$ 为起始状态转移到 $X(t_f)$ 这一终了状态,为了使整个过程(从 $X(t_0)$ 到 $X(t_f)$ )为最优,在 $t_1 < t < t_f$ 间隔内的控制也必须是最优的。这说明最优控制具有无后效性(non-ageing effect),即 $t_1$ 以前的状态不影响 $t_1$ 以后的转移。

马尔可夫随机过程就是具有无后效性过程,最简单的一种情况是:如果某随机过程第 $n+1$ 步的状态 $X(n+1)$ 的条件概率与 $X(0), X(1), X(2), \dots, X(n-1)$ 等状态无关,仅与 $X(n)$ 状态有关。设

$$X(n)=\gamma_i, X(n+1)=\gamma_j,$$

则马尔可夫性质可用下式表示,即:已知 $X(0), X(1), X(2), \dots, X(n-1), X(n)$ (其中 $X(n)=\gamma_i$ )的条件下,出现 $X(n+1)=\gamma_j$ 的条件概率可表示为:

$$\begin{aligned} p_{ij} &= P\{X(n+1)=\gamma_j | X(0), X(1), \dots, X(n-1), X(n)=\gamma_i\} \\ &= P\{X(n+1)=\gamma_j | X(n)=\gamma_i\} \end{aligned} \quad (19-4-1)$$

上式中 $p_{ij}$ 表示从状态 $X(n)=\gamma_i$ 走一步后转移到状态 $X(n+1)=\gamma_j$ 的概率,称为转移概

率。

马尔可夫过程是工程中最常见的随机过程。研究马尔可夫过程主要要解决两个问题：

(1) 设系统现在处于 $\gamma_i$ 状态，从现在起走 $n$ 步以后（或经过 $t$ 时间以后），系统的状态转移到 $\gamma_j$ 状态的概率是多大，从而研究系统的动态特性。

(2) 如果系统原来处于 $\gamma_i$ 状态，当 $t \rightarrow \infty$ （或 $n \rightarrow \infty$ ）时，是否存在一个与 $\gamma_i$ 无关的极限概率 $p_i$ （ $p_i$ 表示系统处于状态 $\gamma_i$ 的概率）。这是研究系统的稳定性问题。

如前所述，第一个问题就是研究马尔可夫过程的转移概率，而第二个问题就是研究马尔可夫过程的遍历性问题。

离散（状态和参数均为离散）的马尔可夫过程又叫做马尔可夫链。下面我们详细看看马尔可夫链的转移概率和遍历性是怎样的。

1. 转移概率，定义式由(19—4—1)式给出的。如果从第 $n$ 步的状态 $X(n)=\gamma_i$ ，走一步到状态 $X(n+1)=\gamma_j$ 的概率为 $p_{ij}$ 与 $n$ 无关，即第多少步对“一步转移概率”无关，数学上表示为：

$$\begin{aligned} P\{X(n+1)=\gamma_j | X(n)=\gamma_i\} &= P\{X(1)=\gamma_j | X(0)=\gamma_i\} \\ &= p_{ij} = \text{常数} \end{aligned} \quad (19-4-2)$$

上式给出这种马尔可夫过程具有时齐性（Time homogeneous），即一步转移概率对时间参数的齐次性。这反映出马尔可夫过程具有稳定的一步转移概率特性。一步稳定转移概率 $p_{ij}$ 具有以下性质：

$$\sum_{j=0}^{N-1} p_{ij} = 1, \quad p_{ij} \geq 0, \quad \text{对所有状态 } X(i) \text{ 均成立。} \quad (19-4-3)$$

(19—4—3)式的物理意义是：从任一状态 $X(n)=\gamma_i$ 出发，经过一次转移后必然到达 $X(n+1)=\gamma_0, \gamma_1, \dots, \gamma_{N-1}$ 中任一状态。

马尔可夫链的转移概率可以用矩阵形式表示，称为随机矩阵或转移概率矩阵：

$$P = \begin{pmatrix} p_{00} & p_{01} & p_{02} & \cdots & p_{0N-1} \\ p_{10} & p_{11} & p_{12} & \cdots & p_{1N-1} \\ p_{20} & p_{21} & p_{22} & \cdots & p_{2N-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{N-10} & p_{N-11} & p_{N-12} & \cdots & p_{N-1N-1} \end{pmatrix} \quad (19-4-4)$$

$P$ 矩阵中每一元素 $p_{ij}$ 有如下特点：

(I)  $0 \leq p_{ij} \leq 1$ ，即各元素非负。

(II)  $\sum_{j=0}^{N-1} p_{ij} = 1$ ，即每行元素之和为1。

根据马尔可夫性质和条件概率定义可证明：第 $n$ 步的转移概率矩阵 $P^{(n)}$ 中的各元素 $p_{ij}^{(n)}$ 为一步转移概率矩阵 $P$ （即(19—4—4)式给出）的 $n$ 次幂中第 $i$ 行第 $j$ 列元素。因此，我们可以用一步转移概率来表示多步转移概率：

$$p_{ij}^{(n)} = \sum_{k=0}^{N-1} p_{ik} p_{kj}^{(n-1)} = \sum_{k=0}^{N-1} p_{ik}^{(n-1)} p_{kj} \quad (19-4-5)$$

(19—4—5)式是马尔可夫链的基本方程，称为Uhapman—Kolmogorov方程。

2. 遍历性

遍历性是指不管初始状态如何, 系统经过相当长时间 (相当多的步数后), 系统处于 $\gamma_i$ 状态的概率为多少。转移概率的遍历性问题, 用数学式子表示为:

$$\lim_{n \rightarrow \infty} p_{ij}^{(n)} = a_i \quad j=0, 1, 2, \dots, N-1$$

这式的物理意义是, 不论从那个状态 $X(n)=\gamma_i$ 出发, 当 $n$ 充分大时, 到达 $\gamma_i$ 状态的概率都接近于 $a_i$ 。反之, 当 $n$ 很大时, 可用 $a_i$ 作为 $p_{ij}^{(n)}$ 的近似值。

$a_i$ 是一个常数, 称为 $n \rightarrow \infty$ 时系统处于 $\gamma_i$ 状态的稳态转移概率。 $a_i$ 存在的条件为: 马尔可夫链具有不可约性, 即从链中任一状态出发都能以某一概率值 ( $>0$ ), 经过有限次转移到链中任何其他状态。就是说, 对任何 $\gamma_i, \gamma_j$ , 存在一个 $n$ 值, 使 $p_{ij}^{(n)} > 0$ 。

下面我们就具体的例子来使用以上所述的基本概念, 来分析一个具体的马尔可夫链。

本例是一个市场管理问题。某种商品有A, B两种商标和其他种商标。经测定这种商品中A种商标的占35%的市场, B种商标的占30%的市场, 而其他商标的占35%的市场。若整个市场作为一个系统来看, 系统的状态这样来定义:

$$X(i) = \begin{cases} 0 & \text{顾客买A种商标的商品} \\ 1 & \text{顾客买B种商标的商品} \\ 2 & \text{顾客买其他种商标的商品} \end{cases}$$

由测定得出的一步转移概率矩阵, 其各元素之值如下表:

表 19-2

开 始 状 态	末 状 态		
	(A) 0	(B) 1	(其他) 2
(A) 0	0.90	0.02	0.08
(B) 1	0.04	0.87	0.09
(其他) 2	0.15	0.12	0.73

一步转移概率矩阵中各元素的物理意义是:

$$p_{00} = \text{顾客刚刚买了A种商标, 下一时刻 (下一步) 又买A种商标的概率} \\ = 0.90$$

$$p_{01} = \text{顾客刚刚买了A种商标, 下一时刻 (下一步) 就买B种商标的概率} \\ = 0.02$$

$$p_{22} = \text{顾客刚刚买了B种商标, 下一时刻 (下一步) 就买其他商标的概率} \\ = 0.09$$

另外其他各元素的物理意义就一目了然了。显然, 随着时间变化 (步数加大), 这就构成一个随机过程, 而且是有限状态的马尔可夫链。

研究这样的马尔可夫链, 就可以确定以下几个问题:

- (I) 在以后各个时刻, 各种商标占有市场规模是如何变化的。
- (II) 如果市场达到一个平衡点, 那么每种商标将有多大的稳定市场。
- (III) A种商标和B种商标的商品所占市场增加和减少的速率。



(IV) 为获得更大的市场, 应采取什么样的措施。

现在来看看, 马尔可夫链分析在成功地解决以上问题时, 是如何进行的。

回答上面第一个问题, 实际上就是求随着时间的每往前推进一步, 各种商标的某种商品占有市场的变化为多大。为此我们首先来看看从第 0 步到第 1 步, A 种商标的某种商品的市场变化为多大。显然下式是成立的:

$$\begin{aligned} \left( \begin{array}{l} \text{在 1 步 A 种商} \\ \text{标占有市场} \end{array} \right) &= \left( \begin{array}{l} \text{在 0 步 A 种商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{l} \text{在 0 步中买 A 种商标} \\ \text{的顾客中下一时刻仍} \\ \text{买 A 种商标的比例数} \end{array} \right) \\ &+ \left( \begin{array}{l} \text{在 0 步 B 种商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{l} \text{在 0 步中买 B 种商标的} \\ \text{顾客中下一时刻要买 A} \\ \text{种商标的比例数} \end{array} \right) \\ &+ \left( \begin{array}{l} \text{在 0 步其他商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{l} \text{在 0 步中买其他商标} \\ \text{的顾客中下一时刻要} \\ \text{买 A 种商标的比例数} \end{array} \right) \end{aligned} \quad (19-4-6)$$

在 (19-4-6) 式中, “在 0 步中买 A 种商标的顾客中下一时刻仍买 A 种商标的比例数”就是在转移矩阵中给出的  $p_{00}$ , 所以上式代入已知数值为:

$$\begin{aligned} \left( \begin{array}{l} \text{在 1 步 A 种商} \\ \text{标占有市场} \end{array} \right) &= (0.35) \cdot p_{00} + (0.30) \cdot p_{10} + (0.35) \cdot p_{20} \\ &= (0.35) \cdot (0.90) + (0.30) \cdot (0.04) + (0.35) \cdot (0.15) \\ &= 0.3795 \end{aligned}$$

同理:

$$\begin{aligned} \left( \begin{array}{l} \text{在 1 步 B 种商} \\ \text{标占有市场} \end{array} \right) &= \left( \begin{array}{l} \text{在 0 步 A 种商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{l} \text{在 0 步买 A 种商标的} \\ \text{顾客中下一时刻要买} \\ \text{B 种商标的比例数} \end{array} \right) \\ &+ \left( \begin{array}{l} \text{在 0 步 B 种商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{l} \text{在 0 步买 B 种商标的} \\ \text{顾客中下一时刻要买} \\ \text{B 种商标的比例数} \end{array} \right) \\ &+ \left( \begin{array}{l} \text{在 0 步其他商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{l} \text{在 0 步买其他商标的} \\ \text{顾客中下一时刻要买} \\ \text{B 种商标的比例数} \end{array} \right) \\ &= (0.35) \cdot p_{01} + (0.30) \cdot p_{11} + (0.35) \cdot p_{21} \\ &= (0.35) \cdot (0.02) + (0.30) \cdot (0.87) + (0.35) \cdot (0.12) \\ &= 0.3100 \end{aligned}$$

$$\begin{aligned} \left( \begin{array}{l} \text{在 1 步其他商} \\ \text{标占有市场} \end{array} \right) &= \left( \begin{array}{l} \text{在 0 步 A 种商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{l} \text{在 0 步买 A 种商标的} \\ \text{顾客中下一时刻要买} \\ \text{其他商标的比例数} \end{array} \right) \\ &+ \left( \begin{array}{l} \text{在 0 步 B 种商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{l} \text{在 0 步买 B 种商标的} \\ \text{顾客中下一时刻要买} \\ \text{其他商标的比例数} \end{array} \right) \end{aligned}$$

$$\begin{aligned}
 & + \left( \begin{array}{c} \text{在 0 步其他商} \\ \text{标占有市场} \end{array} \right) \times \left( \begin{array}{c} \text{在 0 步买其他商标的} \\ \text{顾客中下一时刻仍买} \\ \text{其他商标的比例数} \end{array} \right) \\
 & = (0.35) \cdot p_{02} + (0.30) \cdot p_{12} + (0.35) \cdot p_{22} \\
 & = (0.35) \cdot (0.08) + (0.30) \cdot (0.09) + (0.35) \cdot (0.73) \\
 & = 0.3105
 \end{aligned}$$

因此可以给出下表，列出市场经一步以后的变化结果。

表 19-3

步 序 \ 商 标	A	B	其 他
0	0.3500	0.3000	0.3500
1	0.3795	0.3100	0.3105

那么，若时间再往前推进一步，即求第 2 步后市场变化结果如何呢？还是从具体的 B 种商标说起。可以看出，从 0 步开始，经过 2 步以后，顾客仍要买 B 种商标的事件可以有三种互不相容的途径，即这个人可以：

- (a) 在 0 步买 B 种商标，在 1 步买 A 种商标，在 2 步买 B 种商标；
- (b) 在 0 步买 B 种商标，在 1 步买 B 种商标，在 2 步又买了 B 种商标；
- (c) 在 0 步买 B 种商标，在 1 步买其他商标，在 2 步买 B 种商标。

用图示意如下：

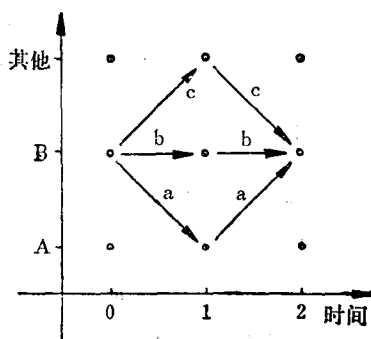


图 19-4-1 从商标 B 经二步后到商标 B 的途径

我们知道：

$$p(a) = p_{10} \cdot p_{01} = (0.04) \cdot (0.02) = 0.0008$$

$$p(b) = p_{11} \cdot p_{11} = (0.87) \cdot (0.87) = 0.7169$$

$$p(c) = p_{12} \cdot p_{21} = (0.09) \cdot (0.12) = 0.0108$$

$$p_{11}^{(2)} = p(a) + p(b) + p(c)$$

$$= 0.0008 + 0.7169 + 0.0108$$

$$= 0.7685$$

$$\text{同理 } p_{20}^{(2)} = p_{20}p_{00} + p_{21}p_{10} + p_{22}p_{20}$$

$$= (0.15) \cdot (0.90) + (0.12) \cdot (0.04) + (0.73) \cdot (0.15)$$

$$= 0.2493$$

∴ 可以得出推导二步后的转移概率矩阵各元素的一般公式为：

$$P_{r,s}^{(2)} = P_{r,0}P_{0,s} + P_{r,1} \cdot P_{1,s} + P_{r,2} \cdot P_{2,s} \quad (19-4-7)$$

(19-4-7)式是(19-4-5)式在 $n=1$ 时的特例。

由(19-4-7)式可以求出全部二步转移概率矩阵的各个元素之值，得出：

$$P^{(2)} = \begin{bmatrix} 0.8228 & 0.0450 & 0.1322 \\ 0.0843 & 0.7685 & 0.1472 \\ 0.2493 & 0.1950 & 0.5557 \end{bmatrix}$$

有了 $P^{(2)}$ ，就可以使用(19-4-6)式，求出二步以后的各种商标占有市场的情况。欲求二步以上各种商标占有市场的变动情况，也同样先求出各步后的转移概率矩阵，再使用(19-4-6)式，求出各种商标占有市场的大小。下表给出从0步到10步市场变动情况。

表 19-4

商 标 步 数	商 标 占 有 市 场		
	A	B	其 他
0	0.35	0.30	0.35
1	0.38	0.31	0.31
2	0.40	0.31	0.28
3	0.42	0.32	0.27
4	0.43	0.32	0.26
5	0.44	0.31	0.25
6	0.44	0.31	0.25
7	0.45	0.31	0.25
8	0.45	0.31	0.24
9	0.45	0.31	0.24
10	0.46	0.31	0.24

相应的求出 $n=2, 3, \dots$ 等的各步转移概率矩阵由下表给出。

表 19-4

步 数	转 移 矩 阵		
2	0.82	0.04	0.13
	0.08	0.77	0.15
	0.25	0.19	0.56
3	0.76	0.07	0.17
	0.13	0.69	0.18
	0.32	0.24	0.44
5	0.67	0.12	0.20
	0.21	0.57	0.22
	0.39	0.29	0.32
10	0.56	0.21	0.23
	0.35	0.41	0.24
	0.45	0.30	0.25

续上表

步 数	转 移 矩 阵		
20	0.49	0.27	0.24
	0.45	0.32	0.24
	0.47	0.29	0.24
30	0.48	0.29	0.24
	0.47	0.30	0.24
	0.47	0.29	0.24
32	0.47	0.29	0.24
	0.47	0.30	0.24
	0.47	0.29	0.24
33	0.47	0.29	0.24
	0.47	0.29	0.24
	0.47	0.29	0.24

从表19-4中可以看到, 经过33步以后处于0状态的概率为0.47, 与0步的初始状态无关, 即  $p_{00}^{(33)} = 0.47$   $i=0, 1, 2$ , 同样  $p_{i1}^{(33)} = 0.29$ ,  $p_{i2}^{(33)} = 0.24$ ,  $i=0, 1, 2$ . 就是说, 每个竞争对象, 经过长时间运行, 各个对象最终所占有的市场大小与它们在0步所占有市场的规模无关。这表明系统达到了一个稳定的状态, 这时的转移概率叫做稳定的转移概率。即有:

$$\lim_{n \rightarrow \infty} p_{rs}^{(n)} = a_s, \quad s=0, 1, 2, \dots, N-1 \quad (19-4-8)$$

稳定状态下的稳定转移概率  $a_s$  之值, 不仅可以象前面所述那样, 一步一步地算出  $p_{rs}^{(n)}$ , 并使  $n \rightarrow \infty$ , 得到  $a_s$  之值; 也可以直接由一步转移概率矩阵  $P$  求出, 此时只要加一个附加条件

$\sum_{s=0}^{N-1} a_s = 1$ 。所以得出联立方程组

$$\begin{cases} a_s = \sum_{r=0}^{N-1} a_r p_{rs} \\ 1 = \sum_{s=0}^{N-1} a_s \end{cases} \quad s=1, 2, \dots, N-1 \quad (19-4-9)$$

(19-4-9) 式就叫做马尔可夫链的稳定状态方程。就我们给出的例题来说, 有:

$$\begin{cases} a_1 = a_0 p_{01} + a_1 p_{11} + a_2 p_{21} \\ a_2 = a_0 p_{02} + a_1 p_{12} + a_2 p_{22} \\ 1 = a_0 + a_1 + a_2 \end{cases}$$

代入一步转移概率矩阵各元素之值后得:

$$\begin{cases} a_1 = (0.02) \cdot a_0 + (0.87) \cdot a_1 + (0.12) \cdot a_2 \\ a_2 = (0.08) \cdot a_0 + (0.09) \cdot a_1 + (0.73) \cdot a_2 \\ 1 = a_0 + a_1 + a_2 \end{cases}$$

解出  $a_0 = 0.4718$ ,  $a_1 = 0.2943$ ,  $a_2 = 0.2369$ 。以上结果与  $p_{i1}^{(33)}$  之值完全相同。可以看

出稳定转移概率之值，其实就是各商标的商品最终占有市场的分配值。

到此为止前面的三个问题我们都已作出回答，那么来看第四个问题，即为获得更大的市场应采取什么样的策略。以B种商标为例，B种商标的商品最终只占有29%的市场，不满意，因此考虑两个修改方案。

计划1：使买B种商标的老主顾增加6%，而使转去买A种商标和其他种商标的顾客减少到2%和5%。则这时的转移概率矩阵为：

$$P = \begin{pmatrix} 0.90 & 0.02 & 0.08 \\ 0.02 & 0.93 & 0.05 \\ 0.15 & 0.12 & 0.73 \end{pmatrix}$$

新的转移概率矩阵各元素都大于0，因此存在稳定转移概率，即各种商标的商品都有一个长期稳定的市场分配，这由稳定状态的方程来解出。

$$\begin{cases} a_1 = 0.02a_0 + 0.93a_1 + 0.12a_2 \\ a_2 = 0.08a_0 + 0.05a_1 + 0.73a_2 \\ 1 = a_0 + a_1 + a_2 \end{cases}$$

解出： $a_0 = 0.3740$ ， $a_1 = 0.4347$ ， $a_2 = 0.1913$ 。

∴在计划1的情况下，B种商标的商品的长期稳定的市场分配已从原来的29%提高到43%。

计划2：商标B的商品制造商力争每步都吸引买其他商标的顾客中的18%，而不是现在的12%。

这时的转移概率矩阵为：

$$P = \begin{pmatrix} 0.90 & 0.02 & 0.08 \\ 0.04 & 0.87 & 0.09 \\ 0.09 & 0.18 & 0.73 \end{pmatrix}$$

其稳定状态方程为：

$$\begin{cases} a_1 = 0.02a_0 + 0.87a_1 + 0.18a_2 \\ a_2 = 0.08a_0 + 0.09a_1 + 0.73a_2 \\ 1 = a_0 + a_1 + a_2 \end{cases}$$

解出 $a_0 = 0.3711$ ， $a_1 = 0.3892$ ， $a_2 = 0.2397$

按照计划1，B种商标的商品最后可以具有43%的市场，而按照计划2，B种商标的商品最后只占有市场的39%。所以，计划1要比计划2好。当然，这两个计划都比原来的安排要好。

## 二、程序说明：

本程序使用P、R、W、M、E、X等数组，并使用F、S、U、Q、N、N<sub>1</sub>等简单变量。

本程序所附例题就是上面所述例题。

## 三、使用本程序的操作方法

1. 用BASIC引入本程序纸带，用RUN键盘命令启动本程序，机器自动印出“INPUT N”，询问系统的状态个数。
2. 用键盘回答系统总的状态个数后，机器又自动印出“INPUT P(N)”，询问系统在0步时处于各种状态的概率值。
3. 用键盘回答系统在0步时处于各状态的概率值后，机器又自动印出“INPUT R(N,N)”，询问系统的一步转移概率矩阵的各元素之值。
4. 用键盘回答一步转移概率矩阵之后，机器自动进行链分析，在运算过程中分别印

出一步转移概率矩阵的 1 次幂, 5 次, 10 次……等各次幂值, 直到每列上下元素之差在小数点后第四位不同为止, 认为系统进入稳定状态, 然后印出达到稳定转移概率值以后每步市场变动情况。

#### 四、程序清单及运行结果

##### 程序清单

```
8000 REM MARKOV CHAINS ANALYSIS
8002 PRINT "INPUT N"
8004 INPUT N
8006 PRINT
8008 DIM P(N), R(N,N), W(N,N), M(N,N), E(N), X(N,N)
8010 DIM T(102,N), A(N,N)
8012 PRINT "INPUT P(N)"
8014 PRINT
8016 FOR I=1 TO N
8018   INPUT P(I)
8020   LET T(0,I)=P(I)
8022 NEXT I
8024 PRINT
8026 PRINT "INPUT R(N,N)"
8028 PRINT
8030 FOR I=1 TO N
8032   FOR J=1 TO N
8034     INPUT R(I,J)
8036     LET W(I,J)=R(I,J)
8038   NEXT J
8040   PRINT
8042 NEXT I
8046 FOR K=1 TO 100
8048   LET N1=0
8050   IF K=1 THEN GOTO 8104
8052   FOR I=1 TO N
8054     FOR J=1 TO N
8056       LET M(I,J)=0
8058       FOR L=1 TO N
8060         LET M(I,J)=M(I,J)+R(I,L)*W(L,J)
8062       NEXT L
8064     NEXT J
8066   NEXT I
8068   FOR I=1 TO N
```

```

8070     FOR J=1 TO N
8072         LET W(I,J)=M(I,J)
8074     NEXT J
8076 NEXT I
8078     FOR J=1 TO N
8080         FOR I=1 TO N-1
8082             LET L=I+1
8084             FOR I1=L TO N
8086                 LET F=W(I,J)-W(I1,J)
8088                 IF ABS(F)>.0001 THEN GOTO 8100
8090             NEXT I1
8092         NEXT I
8094     NEXT J
8096     LET N1=1
8098     GOTO 8104
8100     LET S=K/5
8102     IF S*5><K THEN GOTO 8122
8104     PRINT "MATRIX RAISED TO POWER", K
8106     PRINT
8108     FOR I=1 TO N
8110         FOR J=1 TO N
8112             PRINT W(I,J),
8114         NEXT J
8116     PRINT
8118 NEXT I
8120 PRINT
8122     FOR J=1 TO N
8124         LET E(J)=0
8126         FOR I=1 TO N
8128             LET E(J)=E(J)+P(I)*W(I,J)
8130         NEXT I
8132     NEXT J
8134     LET U=U+1
8136     FOR J=1 TO N
8138         LET T(U,J)=E(J)
8140     NEXT J
8142     IF N1=1 THEN GOTO 8152
8144 NEXT K
8146 PRINT "STEADY-STATE PROBABILITIES HAVE NOT BEEN",
8148 PRINT "OBTAINED IN 100 STEPS"

```

```

8150 GOTO 8288
8152 PRINT "THE STEADY-STATE PROBABILITY OF BEING IN"
8154 PRINT
8156 FOR J=1 TO N
8158     LET Q=J-1
8160     PRINT "STATE": Q; "IS": W(N,J)
8162 NEXT J
8164 PRINT
8166 PRINT "PROBABILITY OF BEING IN STATE J AFTER I STEPS"
8168 PRINT "J=,";"1","2","3","..."
8170 PRINT "I)"
8172 FOR I=0 TO K
8174     PRINT I;
8176     FOR J=1 TO N
8178         PRINT T(I,J),
8180     NEXT J
8182     PRINT
8184 NEXT I
8288 END

```

运行结果

RUN

INPUT N

? 3

INPUT P(N)

? .35 ? .30 ? .35

INPUT R(N,N)

? .9 ? .02 ? .08

? .04 ? .87 ? .09

? .15 ? .12 ? .73

MATRIX RAISED TO POWER 1

.9 .02 .08

.04 .87 .09

.15 .12 .73

MATRIX RAISED TO POWER 5

.6746 .121665 .203734

.209593 .570232 .220174

.390435 .286068 .323495

MATRIX RAISED TO POWER 10

.56013 .209735 .230133

.346871 .413649 .239477



.449649 .303169 .247179  
MATRIX RAISED TO POWER 15

.511674 .253579 .234742  
.414197 .346585 .239214  
.463382 .298293 .23832

MATRIX RAISED TO POWER 20

.489975 .274004 .236015  
.445456 .316458 .238079  
.468166 .29465 .237179

MATRIX RAISED TO POWER 25

.480115 .283375 .236503  
.459786 .302758 .237448  
.470184 .292827 .236982

MATRIX RAISED TO POWER 30

.475618 .287659 .236715  
.466336 .296508 .237147  
.471086 .291977 .236928

MATRIX RAISED TO POWER 35

.473565 .289615 .236811  
.469326 .293655 .237008  
.471496 .291587 .236907

MATRIX RAISED TO POWER 40

.472627 .290507 .236854  
.470692 .292352 .236944  
.471683 .291408 .236898

MATRIX RAISED TO POWER 45

.472199 .290915 .236874  
.471315 .291757 .236915  
.471767 .291326 .236894

MATRIX RAISED TO POWER 50

.472003 .291101 .236883  
.471599 .291485 .236901  
.471806 .291288 .236892

MATRIX RAISED TO POWER 55

.471913 .291185 .236886  
.471728 .29136 .236895  
.471823 .291271 .23689

MATRIX RAISED TO POWER 59

.471877 .291218 .236888  
.471779 .291312 .236892

.471829 .291264 .23689  
 THE STEADY-STATE PROBABILITY OF BEING IN  
 STATE 0 IS .471829  
 STATE 1 IS .291264  
 STATE 2 IS .23689  
 PROBABILITY OF BEING IN STATE J AFTER I STEPS

	J=:1	2	3	...
I)				
0	.35	.3	.35	
1	.3795	.31	.3105	
2	.400525	.31455	.284925	
3	.415793	.31586	.268346	
4	.4271	.315315	.257584	
5	.43564	.313776	.250582	
6	.442214	.311768	.246016	
7	.447366	.309604	.243028	
8	.451467	.307466	.241064	
9	.454779	.305452	.239766	
10	.457484	.303611	.238902	
11	.459715	.301959	.238322	
12	.46157	.300498	.237929	
13	.463122	.299216	.237658	
14	.464427	.298099	.23747	
15	.465529	.297131	.237336	
16	.466461	.296295	.237239	
17	.467253	.295574	.237168	
18	.467925	.294955	.237115	
19	.468498	.294423	.237074	
20	.468986	.293966	.237042	
21	.469402	.293575	.237016	
22	.469757	.29324	.236996	
23	.470061	.292954	.236979	
24	.470319	.292708	.236965	
25	.47054	.292498	.236954	
26	.470729	.292319	.236944	
27	.470891	.292165	.236936	
28	.471029	.292033	.23693	
29	.471146	.291921	.236924	
30	.471247	.291825	.236919	
31	.471333	.291743	.236915	

32	.471407	.291672	.236911
33	.47147	.291612	.236908
34	.471523	.291561	.236906
35	.471569	.291517	.236904
36	.471608	.291479	.236902
37	.471642	.291447	.2369
38	.47167	.29142	.236899
39	.471695	.291396	.236897
40	.471716	.291376	.236896
41	.471734	.291359	.236896
42	.471749	.291344	.236895
43	.471762	.291331	.236894
44	.471773	.291321	.236894
45	.471782	.291311	.236893
46	.47179	.291303	.236893
47	.471797	.291297	.236892
48	.471803	.291291	.236892
49	.471808	.291286	.236891
50	.471813	.291282	.236891
51	.471816	.291278	.236891
52	.471819	.291275	.236891
53	.471822	.291272	.236891
54	.471824	.29127	.23689
55	.471826	.291268	.23689
56	.471827	.291266	.23689
57	.471829	.291264	.23689
58	.47183	.291263	.23689
59	.471831	.291262	.23689

END AT 8288

#### 参考文献

- [1] 蔡宣三,《马尔可夫模型在系统工程中的应用》,清华大学自动化系 1980.
- [2] Leon Cooper, U. Narayana Bhat, Larry J. Lebeance 《Introduction to operations research models》 1977.
- [3] Billy E. Gillett  
《Introduction to operations research a Computer-oriented algorithmic approach》 1976.
- [4] 王梓坤译 E. B. Dynkin 著  
《马尔可夫过程论基础》 1959.

## 第二十章 控制系统仿真

### §1 面向微分方程的数字仿真程序

#### 一、方法概要:

本程序适用于单输入/单输出线性系统的数字仿真。它是将一个 $n$ 阶线性微分方程表示成一组由 $n$ 个一阶微分方程式组成的方程组,而每一个一阶微分方程可以用数值积分法来求解。因此本程序的算法主要包括两部分:一是将 $n$ 阶微分方程转变成一阶微分方程组(状态方程)。二是用四阶龙格-库塔法求解微分方程。现分述于下:

1. 将 $n$ 阶微分方程转变成一阶微分方程组,若系统的微分方程为,

$$\begin{aligned} \frac{d^ny}{dt^n} + a_1 \frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_{n-1} \frac{dy}{dt} + a_n y \\ = c_0 \frac{d^{n-1}u}{dt^{n-1}} + c_1 \frac{d^{n-2}u}{dt^{n-2}} + \cdots + c_{n-1} u \end{aligned}$$

设  $\frac{d}{dt} \triangleq P$  则上式可写成:

$$P^ny + a_1 P^{n-1}y + \cdots + a_n y = c_0 P^{n-1}u + c_1 P^{n-2}u + \cdots + c_{n-1} u$$

$$\text{即: } \sum_{j=1}^n a_{n-j+1} P^j y = \sum_{j=0}^{n-1} c_{n-j-1} P^j u \quad (20-1-1)$$

引进 $n$ 个状态变量:  $x_1, x_2, \dots, x_n$

$$\text{令: } \begin{cases} u = \sum_{j=1}^n a_{n-j+1} P^j x \\ P^j x = x_{j+1} \quad (j=0, 1, 2, \dots, n-1) \end{cases} \quad (20-1-2)$$

(即:  $x_1 = x; x_2 = \dot{x}_1; x_3 = \ddot{x}_1 = \dot{x}_2; \dots, x_{n-1} = \dot{x}_{n-2}; x_n = \dot{x}_{n-1}$ )

$$\therefore \sum_{j=0}^{n-1} a_{n-j-1} x_{j+1} + a_n P^n x = u$$

若 $a_n = 1$ 得:

$$\begin{cases} P^n x_1 = \dot{x}_n = \sum_{j=0}^{n-1} a_{n-j-1} x_{j+1} + u \\ \dot{x}_{n-1} = x_n \\ \dot{x}_{n-2} = x_{n-1} \\ \vdots \\ \dot{x}_1 = x_2 \end{cases} \quad (20-1-3)$$

写成矩阵形式状态方程为,

$$\begin{pmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \\ -a_n & -a_{n-1} & \cdots & \cdots & \cdots & -a_1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} u$$

$$\dot{x} = Ax + Bu$$

将式 (20-1-2) 代入式 (20-1-1) 得:

$$\sum_{j=1}^n a_{n-j} p^j y = \sum_{j=0}^{n-1} c_{n-j-1} p^j u = \sum_{j=0}^{n-1} c_{n-j-1} p^j \cdot \sum_{i=1}^n a_{n-i} p^i x$$

$$y = \sum_{j=0}^{n-1} C_{n-j-1} p^j x = \sum_{j=0}^{n-1} C_{n-j-1} x_{j+1} = Cx$$

$$y = (c_{n-1}, c_{n-2}, \cdots, c_0) \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \\ -a_n & -a_{n-1} & \cdots & \cdots & \cdots & -a_1 \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad C = (c_{n-1}, c_{n-2}, \cdots, c_0)$$

## 2. 四阶龙格—库塔法计算公式:

设系统的微分方程组为:

$$\begin{cases} \frac{dx_1}{dt} = f_1(t, x_1, \cdots, x_n) \\ \frac{dx_2}{dt} = f_2(t, x_1, \cdots, x_n) \\ \cdots \cdots \cdots \\ \frac{dx_n}{dt} = f_n(t, x_1, \cdots, x_n) \end{cases}$$

四阶龙格—库塔法公式为: (以第  $i$  个状态量为例)

$$\begin{cases} x_i^{(1)} = x_i^{(0)} + \frac{h}{6}(K_{i1} + 2K_{i2} + 2K_{i3} + K_{i4}) \\ K_{i1} = f_i(t_0, x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}) \\ K_{i2} = f_i(t_0 + \frac{h}{2}, x_1^{(0)} + \frac{h}{2}K_{i1}, \dots, x_n^{(0)} + \frac{h}{2}K_{n1}) \\ K_{i3} = f_i(t_0 + \frac{h}{2}, x_1^{(0)} + \frac{h}{2}K_{i2}, \dots, x_n^{(0)} + \frac{h}{2}K_{n2}) \\ K_{i4} = f_i(t_0 + h, x_1^{(0)} + hK_{i3}, \dots, x_n^{(0)} + hK_{n3}) \end{cases}$$

本线性系统中

$$\dot{x}_i = \frac{dx_i}{dt} = a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n + b_i u$$

$$x_i^{(1)} = x_i^{(0)} + \frac{h}{6}(K_{i1} + 2K_{i2} + 2K_{i3} + K_{i4})$$

其中,  $x_i^{(1)}$  表示  $t=t_1=t_0+h$  时的  $x_i$  值。

$x_i^{(0)}$  表示  $t=t_0$  时的  $x_i$  值

## 二、程序说明:

1. 本程序包含以下几个程序块: 初始程序块, 输入程序块, 运算程序块, 输出程序块。

初始程序块主要对所需的数组定维、工作单元置初值。(语句4—42)

输入程序块是输入系统参数初值, 计算步距, 打印间隔, 打印点数。(语句46—76)

运行程序块用来求解系统的动态响应, 是本程序的核心。(语句78—165)

输出程序块用来输出计算结果, 可根据需要由电传打出数据, 或用  $x-y$  记录仪或图像显示仪画出曲线。(语句167—180)

2. 程序使用工作单元:

N 系统阶次

U 系统输入阶跃函数的幅度。

$Q_1$  计算步长。

$Q_2$  打印间隔 (每隔  $Q_1 * Q_2$  打印一点)

$Q_3$  打印次数

$N_6$  输出形式标志 ( $N_6=0$  时为电传输出,  $N_6=1$  时为  $x-y$  或显示输出)

$N_7$  时间比列尺 (X坐标)

$N_8$  变量比列尺 (Y坐标)

V 系统反馈系数

$Z(N)$  存放系统参数  $a_1 \dots a_n$

$C(N)$  存放系统参数  $c_0 \dots c_{n-1}$  (即状态方程中  $e$  阵内容)

$A(N, N)$  状态方程中  $A$  阵

$B(N)$  状态方程中  $B$  阵

$H(4)$  龙格—库塔法中  $h_i$  系数

$K(N, 4)$  龙格—库塔法中  $K_{i1}$  系数

P(N,4) 工作单元

X(N) 状态变量  $x_1 \cdots x_n$

D(N,4) 工作单元

### 三、操作说明:

1. 将所计算系统按本程序要求的微分方程形式写出, 其系数是用数据语句给出, 其输入次序为  $a_n, a_{n-1}, \cdots, a_1, c_{n-1}, c_{n-2}, \cdots, c_0$ , 最后输入状态量初值  $x(1) \cdots x(N)$ 。输入语句标号从400开始。(详细请参看试题)。

2. 由BASIC引入源程序, 键盘命令RUN↵, 启动本程序机器顺次印出要求赋值的变量名  $N, Q_1, N_0, Q_2, Q_3, U_0$ 。(若  $N_0=1$  机器还打印出  $N_7, N_8$ )。使用者用键盘顺次回答以上各变量的数值后, 机器自动进行计算, 打印出所需结果。

### 四、使用注意事项:

1. 本程序只适用于  $y, u$  及其各阶导数的初值为 0 时的情况, 即相当传递函数为

$$G(s) = \frac{c_0 s^{n-1} + c_1 s^{n-2} + \cdots + c_{n-1}}{s^n + a_1 s^{n-1} + \cdots + a_n}$$

若系统  $y, u$  及其各阶导数有初值时, 程序要作一些修改才能满足。

2. 本程序要求输入量  $u$  的阶次必须低于输出量  $y$  的阶次, 即传递函数是严格真有理分式, 此系统是物理可实现的。

3. 本程序中输入  $u$  是阶跃函数, 若  $u$  不是阶跃函数, 那么事先要将  $u(t)$  输入, 程序要作如下修改:

增加 305  $T_1 = T + H(J)$

修改 355  $LET K(I, J) = D(I, J) + B(I) * U(T_1)$

### 五、试题:

系统结构图为:

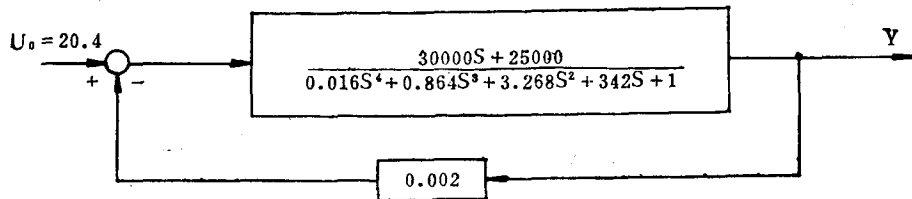


图20-1-1

已知: 
$$W(s) = \frac{3000s + 25000}{0.016s^4 + 0.864s^3 + 3.268s^2 + 3.42s + 1}$$

$$= \frac{1.875 \times 10^6 S + 1.562 \times 10^6}{S^4 + 54S^3 + 204.2S^2 + 213.8S + 62.5}$$

$$a_4 = 62.5, a_3 = 213.8, a_2 = 204.2, a_1 = 54$$

$$c_3 = 1.562 \times 10^6, c_2 = 1.875 \times 10^6, c_1 = 0, c_0 = 0$$

数据语句:

400 DATA 62.5, 213.8, 204.2, 54

401 DATA 1.562E+6, 1.875E+6, 0, 0, 0.002

402 DATA 0, 0, 0, 0

### 五、程序清单及运行结果

程序清单

```

4  PRINT "N"
6  INPUT N
8  PRINT
10 DIM H( 4 ),K(N,4),X(N),P(N,4),Z(N),D(N,4),A(N,N),B(N),C(N)
14  FOR I=0 TO N
16      LET B( I )=0
18      LET C( I )=0
20      LET Z( I )=0
26      FOR J=0 TO 4
28          LET K(I,J)=0
30          LET P(I,J)=0
32      NEXT J
34      FOR J=0 TO N
36          LET A(I,J)=0
40      NEXT J
42  NEXT I
46  PRINT "Q1"
48  INPUT Q1
50  PRINT
52  PRINT "N 5"
54  INPUT N5
56  PRINT
58  IF N5=0 GOTO 66
60  PRINT "N 7,N 8"
62  INPUT N7,N8
64  PRINT
66  PRINT "Q 2,Q 3"
68  INPUT Q2,Q3
69  PRINT
70  PRINT "U"
71  INPUT U
72  PRINT
73  FOR I=1 TO N
74      READ Z( I )
76  NEXT I
78  FOR I=1 TO N
80      FOR J=1 TO N
82          IF I=N GOTO 90
84          IF J=I+1 GOTO 86

```



```

85      GOTO 92
86      LET A(I,J)=1
88      GOTO 92
90      LET A(I,J)=-Z(J)
92      NEXT J
94      NEXT I
96      FOR I=1 TO N
98          READ C(I)
100     NEXT I
101     READ V
102     LET B(N)=1
104     LET I=N
106     FOR J=1 TO N
108         LET A(I,J)=A(I,J)-V*C(J)
110     NEXT J
112     LET H(1)=0
113     LET H(2)=Q1/2
114     LET H(3)=Q1/2
115     LET H(4)=Q1
116     FOR I=1 TO N
117         READ X(I)
118     NEXT I
120     IF N5=1 GOTO 182
122     PRINT "T",TAB(10),"Y"
124     GOSUB 200
126     PRINT T; TAB(10); Y
127     LET T=0
130     FOR E=1 TO Q3
135         FOR G=1 TO Q2
140             GOSUB 300
145             FOR I=1 TO N
150                 LET X(I)=X(I)+Q1*(K(I,1)+2*K(I,2)+2*K(I,3)+K(I,4))/6
155             NEXT I
160             LET T=T+Q1
162             GOSUB 200
165     NEXT G
167     IF N5=1 GOTO 186
170     PRINT T; TAB(10); Y
175     NEXT E
180     END

```

```

182 PRINT "Y0=";Y
183 LET Y2=Y
184 GOTO 130
185 LET Y4=0
186 LET Y3=(Y-Y2)*N8
188 LET Y1=INT (Y3+Y4)
190 LET Y2=Y
192 LET Y4=Y3-Y1+Y4
194 PLOT 1,N7,Y1
196 GOTO 175
200 REM OUTPUT COMPUTATIONS SUBROUTINE
205 LET Y=0
210 FOR I=1 TO N
215   LET Y=C(I)*X(I)+Y
220 NEXT I
225 RETURN
300 REM INTEGRATE SUBROUTINE
302 FOR J=1 TO 4
310   FOR I=1 TO N
315     LET P(I,J)=H(J)*K(I,J-1)
320     LET Z(I)=P(I,J)+X(I)
325   NEXT I
330   FOR I=1 TO N
335     LET D(I,J)=0
340     FOR L=1 TO N
345       LET D(I,J)=D(I,J)+A(I,L)*Z(L)
350     NEXT L
355     LET K(I,J)=D(I,J)+B(I)*U
360   NEXT I
365 NEXT J
371 RETURN
400 DATA 62.5, 213.8, 204.2, 54
401 DATA 1.562E+6, 1.875E+6, 0, 0, .002
402 DATA 0, 0, 0, 0

```

运行结果

RUN

N

? 4

Q1

? .02

```

N5
? 0
Q2, Q3
? 4? 20
U
? 20.4
T          Y
0          0
.08       1400.4
.16       6064.66
.24       11823.8
.32       16079.5
.4        17196.4
.48       15087.4
.559999   11057.6
.639999   7072.37
.719999   4833.38
.799999   5082.43
.879999   7403.35
.959999   10544.3
1.04      13058.2
1.12      13946.5
1.2       13031.3
1.28      10929.3
1.36      8688.7
1.44      7286.53
1.52      7219.13
1.6       8349.33
END AT 180

```

## §2 连续系统结构图法数字仿真程序

### 一、方法概要:

本程序可用来对单输入的线性系统进行仿真，其方法是将系统看作由许多典型环节组成的，将各环节的参数以及各环节相互的连接方式输入给计算机，让程序来闭环系统，建立微分方程组，然后再用数值积分法求解，称为连续系统结构图法数字仿真。

这种方法适用于研究某个参数对系统的影响，也便于研究具有非线性环节的系统。

#### (1) 典型环节的选定:

在控制系统中常见的环节是

积分环节:  $\frac{K}{S}$

比例积分环节:  $\frac{K_1 S + K_2}{S}$

惯性环节:  $\frac{K}{TS+1}$

一阶领先(或迟后)环节:  $K \frac{T_1 S + 1}{T_2 S + 1}$

二阶振荡环节:  $\frac{K}{TS^2 + 2\xi TS + 1}$

对于积分环节、比例积分环节、惯性环节、一阶领先或迟后环节,用  $\frac{C+DS}{A+BS}$  是很容易表示的。

对于二阶振荡环节可用两个  $\frac{C+DS}{A+BS}$  串联, 并加上一个负反馈可得到。如下图所示:

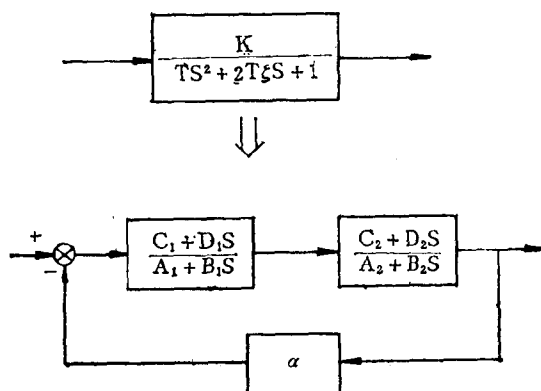


图 20-2-1

其中  $A_1=1$ ,  $B_1=\frac{T}{2\xi}$ ,  $C_1=1$ ,  $D_1=0$

$A_2=0$ ,  $B_2=2T\xi$ ,  $C_2=1$ ,  $D_2=0$

$\alpha = \frac{1}{K}$

(2) 典型环节构成的系统状态方程:

$$\text{已知: } Y_i = \frac{C_i + D_i S}{A_i + B_i S} U_i$$

$$\text{即 } (A_i + B_i S) Y_i = (C_i + D_i S) U_i$$

其中  $i=1, 2, \dots, n$ 。

例如下面图中所示的系统, 其中用方框框起来, 并标上 1, 2, ... 符号的是典型环节, 标上  $\alpha_2, \alpha_4, \beta_4 \dots$  是比例环节, 每个环节的输入为  $V_i$ , 输出为  $Y_i$ , 其运动方程如下:

$$(A_1 + B_1 S) Y_1 = (C_1 + D_1 S) U_1$$

$$(A_2 + B_2 S) Y_2 = (C_2 + D_2 S) U_2$$

$$(A_3 + B_3 S) Y_3 = (C_3 + D_3 S) U_3$$

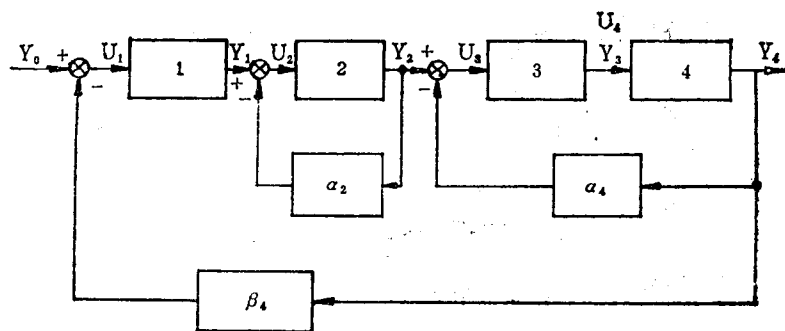


图 20-2-2

$$(A_4 + B_4 S) Y_4 = (C_4 + D_4 S) U_4$$

$$U_1 = Y_0 - \beta_4 Y_4$$

$$U_2 = Y_1 - \alpha_2 Y_2$$

$$U_3 = Y_2 - \alpha_4 Y_4$$

$$U_4 = Y_3$$

写成矩阵形式:

$$(A + BS) Y = (C + DS) U \quad (20-2-1)$$

$$U = W_1 Y + W_0 Y_0 \quad (20-2-2)$$

其中

$$A = \begin{bmatrix} A_1 & 0 \\ & A_2 \\ & & A_3 \\ 0 & & & A_4 \end{bmatrix}$$

$$B = \begin{bmatrix} B_1 & 0 \\ & B_2 \\ & & B_3 \\ 0 & & & B_4 \end{bmatrix}$$

$$C = \begin{bmatrix} C_1 & 0 \\ & C_2 \\ & & C_3 \\ 0 & & & C_4 \end{bmatrix}$$

$$D = \begin{bmatrix} D_1 & 0 \\ & D_2 \\ & & D_3 \\ 0 & & & D_4 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} 0 & 0 & 0 & -\beta_4 \\ 1 & -\alpha_2 & 0 & 0 \\ 0 & 1 & 0 & -\alpha_4 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$W_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix}$$

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

$W_0$ ,  $W_1$ 称为连接矩阵,  $W_0$ 表示输入量作用于各环节的情况,  $W_1$ 表示各环节相互间的连接情况。

将 (20-2-2) 式代入 (20-2-1) 式:

$$(A+BS)Y = (C+DS)(W_1Y+W_0Y_0)$$

$$(B-DW_1)\dot{Y} = (CW_1-A)Y + CW_0\dot{Y}_0 + DW_0\dot{Y}_0$$

$$S\dot{Y} = RY + V_1Y_0 + V_2\dot{Y}_0$$

$$S = B - D * W_1$$

$$R = C * W_1 - A$$

$$V_1 = C * W_0$$

$$V_2 = D * W_0$$

如果 S 阵有逆阵存在, 那么两边左乘  $S^{-1}$ , 则:

$$\dot{Y} = S^{-1}RY + S^{-1}V_1Y_0 + S^{-1}V_2\dot{Y}_0 \quad (20-2-3)$$

设  $Y_0$  为阶跃函数, 且外加作用函数所作用的环节  $D_1=0$ , 即  $D * W_0=0$

则

$$\dot{Y} = S^{-1}RY + S^{-1}V_1Y_0 \quad (20-2-4)$$

此为标准的一阶常微分方程组, 利用第廿章§1程序的方法, 即用四阶龙格库塔法求解。

(3) 连接矩阵  $W_0$ 、 $W_1$  的输入方式:

将连接矩阵  $W_0$ 、 $W_1$  合为一个矩阵, 总称为  $W_1$  是  $n \times (n+1)$  长方形阵,  $W_0$  为  $W$  的第 0 列,  $W_1$  为去掉第零列后的  $n \times n$  方阵。

由于  $W$  阵中零元素较多, 所以在输入时, 将有数值的元素输入, 是零的元素不进行输入。用  $W_{ij}$  表示连接情况,  $i$  表示第  $i$  个环节 (受作用的环节)  $j$  表示第  $j$  个环节对第  $i$  个环节的作用。当  $j=0$  即表示系统的输入对第  $i$  个环节的作用。

对于前例情况:

i	j	$W_{ij}$
1	0	1
1	4	$-\beta_4$
2	1	1
2	2	$-\alpha_2$
3	2	1
3	4	$-\alpha_4$
4	3	1
0	0	0

最后一行 0, 0, 0 是 W 阵数据结束标志。

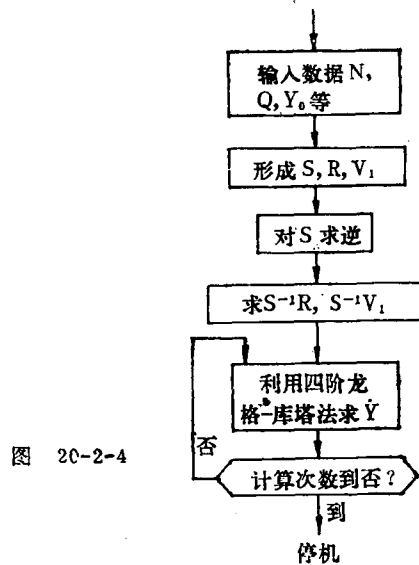
本程序规定连接矩阵数据输入从语句标号 1050 开始, 如上例写成:

1050 DATA 1, 0, 1, 1, 4,  $-\beta_1$ , 2, 1, 1, 2, 2,  $-\alpha_2$ , 3, 2, 1, 3, 4,  $-\alpha_4$ , 4,  
3, 1, 0, 0, 0

(4) S 阵求逆:

采用高斯——约旦法对矩阵求逆, 其方法详见第十四章。

5. 程序框图:



## 二、程序说明:

(1) 为了保证 S 阵有逆存在, 系统中的典型环节不得包括纯比例环节和纯微分环节。否则 S 阵将出现零列, 不得求逆, 因此在有纯比例环节时, 可将其并入下一环节, 或归在 W 阵中。有纯微分环节时, 要将系统结构图进行变换后再使用此程序。

(2) 在程序中输入量规定为阶跃函数, 且输入量加入的环节 D 必须为 0, 即保证  $DW_0 = 0$ , 否则公式 (20-2-3) 中,  $DW_0 \dot{Y}_0$  一项必须考虑, 本程序须进行改造才能使用。

(3) 在程序中未将非线性环节编入, 当系统有非线性环节时, 可遵照离散相似法数字仿真程序 (见本章第 3 节) 进行改造本程序。

(4) 程序中使用的部分变量:

- N 系统环节数目
- $Y_0$  系统输入阶跃函数的幅度
- $Q_1$  计算步长
- $Q_2$  打印间隔 (即每隔  $Q_1 * Q_2$  打印一点)
- $Q_3$  打印次数
- $N_0$  打印变量个数
- $N_2$  } 打印变量编号
- $N_3$  }
- $N_4$  }

### 三、操作说明:

- (1) 画出以典型环节为标准的系统结构图, 并标出环节序号: I, II, III.....
- (2) 写出系统各环节的参数, 由语句1000号开始, 语句按环节的顺序 I, II, III.....排列, N个环节有N个语句。

1000 DATA A(1), B(1), C(1), D(1), Y(1)

1001 DATA A(2), B(2), C(2), D(2), Y(2)

⋮

其中Y(1), Y(2)....为该环节状态变量初始值。

- (3) 写出系统连接情况数据:

从语句标号1050开始, 若W阵为如下数据

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

则写为:

1050 DATA 1, 0, 1, 1, 4, -1, 2, 1, 1,  
3, 2, 1, 4, 3, 1, 0, 0, 0

最后三个零表示结束标志。

- (4) 上机操作:

- ① 输入BASIC解释程序
- ② 输入本仿真程序
- ③ 用电传打字机输入环节参数和连接矩阵。
- ④ 用电传机输入命令RUN↵
- ⑤ 用电传机回答二个问话语句:
  - a) N, Y<sub>0</sub>, Q<sub>1</sub>, Q<sub>2</sub>, Q<sub>3</sub>
  - b) N<sub>0</sub>, N<sub>2</sub>, N<sub>3</sub>, N<sub>4</sub>

### 四、试题

仿真一个小功率随动系统, 其框图如下:

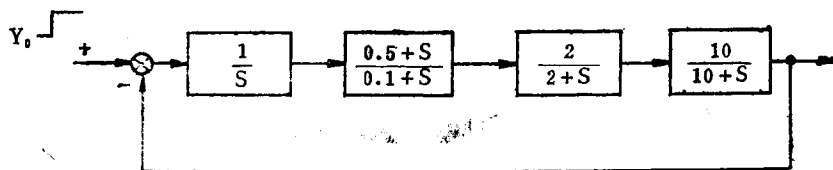


图 20-2-5

系统共有四个环节, 设Y<sub>0</sub>=10, 各环节的状态变量初值为0。

1000 DATA 0, 1, 1, 0, 0

1001 DATA 0.1, 1, 0.5, 1, 0

1002 DATA 2, 1, 2, 0, 0

1003 DATA 10, 1, 10, 0, 0



1050 DATA 1, 0, 1, 1, 4, -1, 2, 1, 1,

3, 2, 1, 4, 3, 1, 0, 0, 0

回答电传三个问题,

$N, Y_0, Q_1, Q_2, Q_3$   
? 4? 10? 0.02? 10? 50

{ 计算步长  $Q_1=0.02$   
打印间隔  $Q_1 * Q_2=0.1$   
打印次数  $Q_3=50$

$N_0, N_2, N_3, N_4$   
? 1? 4? 0? 0

{ 打印第四环节二个输出变量

## 五、程序清单及运行结果

### 程序清单

```
5 PRINT "N,Y0,Q1,Q2,Q3"
7 INPUT N,Y0,Q1,Q2,Q3
10 PRINT
12 PRINT"N0,N2,N3,N4"
13 INPUT N0,N2,N3,N4
14 PRINT
15 PRINT
20 DIM A(N,N),B(N),C(N),D(N,N+5),W(N,N+1)
25 DIM F(N,N),G(N,2*N),S(N,N),R(N,N)
26 DIM V(N),E(N,N),Y(N)
27 FOR I=1 TO N
28 READ A(I,1),B(I),C(I),D(I,1),Y(I)
29 NEXT I
90 FOR I=1 TO N
92 FOR J=1 TO N+1
94 LET W(I,J)=0
96 NEXT J
98 NEXT I
100 READ I1,J1,W1
105 IF (I1+J1)=0 GOTO 135
110 IF J1=0 GOTO 125
115 LET W(I1,J1)=W1
120 GOTO 100
125 LET J1=N+1
130 GOTO 115
135 FOR I=1 TO N
140 LET V(I)=C(I)*W(I,N+1)
145 NEXT I
155 FOR I=1 TO N
160 FOR J=1 TO N
```

```

165         IF I=J GOTO 180
170         LET S(I,J)=0
175     NEXT J
178     GOTO 190
180     LET S(I,J)=B(I)
185     NEXT J
190 NEXT I
195 FOR I=1 TO N
200     FOR J=1 TO N
205         LET E(I,J)=D(I,1)*W(I,J)
210     NEXT J
215 NEXT I
220 FOR I=1 TO N
225     FOR J=1 TO N
230         LET S(I,J)=S(I,J)-E(I,J)
235     NEXT J
240 NEXT I
255 FOR I=1 TO N
260     FOR J=1 TO N
265         IF I=J GOTO 280
270         LET R(I,J)=0
275     NEXT J
278     GOTO 290
280     LET R(I,J)=A(I,1)
285     NEXT J
290 NEXT I
295 FOR I=1 TO N
300     FOR J=1 TO N
305         LET E(I,J)=C(I)*W(I,J)
310     NEXT J
315 NEXT I
320 FOR I=1 TO N
325     FOR J=1 TO N
330         LET R(I,J)=E(I,J)-R(I,J)
335     NEXT J
340 NEXT I
380 GOSUB 701
381 GOTO 400
382 FOR I=1 TO N
383     FOR J=1 TO N

```

```

384     PRINT F(I,J),
385     NEXT J
386     PRINT
387 NEXT I
388 STOP
400 FOR I=1 TO N1
405     FOR J=1 TO N1
410         LET A(I,J)=0
415         FOR K1=1 TO N1
420             LET A(I,J)=A(I,J)+F(I,K1)*R(K1,J)
425         NEXT K1
430     NEXT J
435 NEXT I
450 FOR I=1 TO N1
455     LET B(I)=0
460     FOR J=1 TO N1
465         LET B(I)=B(I)+F(I,J)*V(J)
470     NEXT J
475 NEXT I
500 DIM K(N,5),H(4),P(N,5),Z(N)
525 FOR I=1 TO N1
530     LET K(I,0)=0
535 NEXT I
540 LET H(1)=0
545 LET H(2)=Q1/2
550 LET H(3)=Q1/2
555 LET H(4)=Q1
560 GOTO 1450
565 FOR E1=1 TO Q3
570     FOR G1=1 TO Q2
575         GOSUB 600
580         FOR I=1 TO N1
582             LET Y(I)=Y(I)+Q1*(K(I,1)+2*K(I,2)+2*K(I,3)+K(I,4))/6
584         NEXT I
586         LET T=T+Q1
588     NEXT G1
590     GOTO 1487
592 NEXT E1
594 STOP
600 FOR J=1 TO 4

```

```

605   LET T1=T+H(J)
610   FOR I=1 TO N1
615       LET P(I,J)=H(J)*K(I,J-1)
620       LET Z(I)=Y(I)
625       LET Z(I)=P(I,J)+Z(I)
630   NEXT I
635   FOR I=1 TO N1
640       LET D(I,J)=0
645       FOR L=1 TO N1
650           LET D(I,J)=D(I,J)+A(I,L)*Z(L)
655       NEXT L
660       LET K(I,J)=D(I,J)+B(I)*Y0
665   NEXT I
670 NEXT J
675 RETURN
701 LET N1=N
702 LET F1=N1+1
704 LET H1=N1+N1
708 FOR I=1 TO N1
710     FOR J=1 TO N1
712         LET G(I,J)=S(I,J)
714     NEXT J
716 NEXT I
718 FOR I=1 TO N1
720     FOR J=F1 TO H1
722         LET G(I,J)=0
724     NEXT J
726 NEXT I
728 FOR I=1 TO N1
730     LET J=I+N1
732     LET G(I,J)=1
734 NEXT I
736 FOR K1=1 TO N1
738     LET C1=K1+1
740     IF K1=N1 GOTO 764
742     LET C2=K1
744     FOR I=C1 TO N1
746         IF ABS (G(I,K1))<=ABS (G(C2,K1)) GOTO 750
748         LET C2=I
750     NEXT I

```

```

752   IF C2=K1 GOTO 764
754   FOR J=K1 TO H1
756       LET C3=G(K1,J)
758       LET G(K1,J)=G(C2,J)
760       LET G(C2,J)=C3
762   NEXT J
764   FOR J=C1 TO H1
766       LET G(K1,J)=G(K1,J)/G(K1,K1)
768   NEXT J
770   IF K1=1 GOTO 785
772   LET C4=K1-1
774   FOR I=1 TO C4
776       FOR J=C1 TO H1
778           LET G(I,J)=G(I,J)-G(I,K1)*G(K1,J)
780       NEXT J
782   NEXT I
784   IF K1=N1 GOTO 790
785   FOR I=C1 TO N1
786       FOR J=C1 TO H1
787           LET G(I,J)=G(I,J)-G(I,K1)*G(K1,J)
788       NEXT J
789   NEXT I
790   NEXT K1
791   FOR I=1 TO N1
792       FOR J=1 TO N1
793           LET K1=J+N1
794           LET F(I,J)=G(I,K1)
795       NEXT J
796   NEXT I
797   RETURN
1000  DATA 0, 1, 1, 0, 0
1001  DATA .1, 1, .5, 1, 0
1002  DATA 2, 1, 2, 0, 0
1003  DATA 10, 1, 10, 0, 0
1050  DATA 1, 0, 1, 1, 4, -1, 2, 1, 1, 3, 2, 1, 4, 3, 1, 0, 0, 0
1455  IF N0=3 GOTO 1480
1456  IF N0=2 GOTO 1470
1460  PRINT "T"; TAB (10); "Y("; N2; ")"
1463  GOTO 1484
1470  PRINT "T"; TAB (10); "Y("; N2; ")",

```

```

1471 PRINT TAB (20);"Y(";N3;")"
1473 GOTO 1484
1480 PRINT "T;"TAB (10);"Y(";N2;")";
1484 GOTO 565
1487 IF N0=2 GOTO 1490
1488 PRINT T; TAB (10);Y(N2)
1489 GOTO 1493
1490 PRINTT; TAB (10);Y(N2); TAB (20);Y(N3)
1491 GOTO 1493
1492 PRINT T; TAB (10);Y(N2); TAB (20);Y(N3); TAB (30);Y(N4)
1493 GOTO 592

```

运行结果

RUN

N, Y0, Q1, Q2, Q3

? 4? 10? .02? 10? 50

N0, N2, N3, N4, N5

? 1? 4? 0? 0? 0

T	Y ( 4 )
.2	.158912
.4	.834777
.599999	1.97316
.799999	3.41942
.999999	5.02747
1.2	6.67457
1.4	8.26292
1.6	9.71877
1.8	10.9906
2	12.0466
2.2	12.8723
2.4	13.4672
2.6	13.8423
2.8	14.0168
3	14.016
3.2	13.8687
3.4	13.6051
3.6	13.2554
3.8	12.8481
4	12.4092
4.19999	11.9615
4.39999	11.5243

4.59998	11.1128
4.79998	10.7385
4.99997	10.4095
5.19997	10.1302
5.39996	9.90263
5.59996	9.72594
5.79995	9.59756
5.99995	9.51335
6.19994	9.46815
6.39994	9.45619
6.59993	9.47137
6.79993	9.50766
6.99992	9.5593
7.19992	9.62099
7.39991	9.68802
7.59991	9.75641
7.7999	9.82284
7.9999	9.88475
8.19988	9.94028
8.39987	9.98818
8.59985	10.0278
8.79984	10.059
8.99982	10.0819
9.19981	10.0972
9.3998	10.1056
9.59978	10.108
9.79977	10.1054
9.99975	10.0989

STOP AT 594

### §3 连续系统离散相似法数字仿真程序

#### 一、方法概要:

由于计算机控制技术的广泛应用,将连续系统离散化后进行计算机仿真也得到发展。离散相似法数字仿真是在连续系统的输入、输出端加上虚构的采样开关(离散化),同时为了使输入信号恢复到原来的信号,因此在输入端还要加一个保持器。

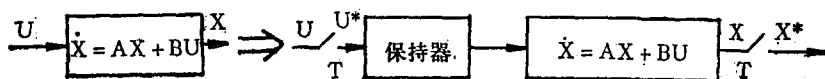


图 20-3-1

本仿真程序把系统分成若干个典型环节,并具有处理非线性环节的功能,使用者只要事先给出各典型环节的参数、各环节之间的连接情况以及注明那些环节具有何种非线性,即能方便地进行处理。

(1) 离散化状态方程的解:

设连续系统的状态方程为  $\dot{X} = AX + BU$  (20-3-1)

经拉氏变换后得:

$$sX(s) - X(0) = AX(s) + BU(s)$$

$$(sI - A)X(s) = X(0) + BU(s)$$

$$X(s) = (sI - A)^{-1}X(0) + (sI - A)^{-1}BU(s)$$

令  $L^{-1}[(sI - A)^{-1}] = \Phi(t) = e^{At}$  称为系统转移矩阵

$$\therefore X(s) = L[\Phi(t)]X(0) + L[\Phi(t)]BU(s)$$

对上式进行拉氏反变换得:

$$X(t) = \Phi(t)X(0) + \int_0^t \Phi(t-\tau)BU(\tau)d\tau \quad (20-3-2)$$

上式为连续系统状态方程的解,由此出发可推导离散化后状态方程的解。

设系统在  $k$  及  $k+1$  两个相邻采样瞬时有:

$$X(kT) = e^{AkT}X(0) + \int_0^{kT} e^{A(kT-\tau)}BU(\tau)d\tau \quad (20-3-3)$$

$$X((k+1)T) = e^{A(k+1)T}X(0) + \int_0^{(k+1)T} e^{A((k+1)T-\tau)}BU(\tau)d\tau \quad (20-3-4)$$

(20-3-4) -  $e^{AT} \times$  (20-3-3) 得:

$$X((k+1)T) = e^{AT}X(kT) + \int_{kT}^{(k+1)T} e^{A((k+1)T-\tau)}BU(\tau)d\tau \quad (20-3-5)$$

由于 (20-3-5) 式右端积分与  $k$  无关,且  $k$  与  $k+1$  之间  $U$  值保持不变

$\therefore U(\tau) = U(kT)$

$$\begin{aligned} X((k+1)T) &= e^{AT}X(kT) + \left[ \int_0^T e^{A(T-\tau)}Bd\tau \right] U(kT) \\ &= \Phi(T)X(kT) + \left[ \int_0^T \Phi(T-\tau)Bd\tau \right] U(kT) \end{aligned}$$

$$\text{令 } \Phi_m(T) = \int_0^T \Phi(T-\tau)Bd\tau$$

$$\therefore X((k+1)T) = \Phi(T)X(kT) + \Phi_m(T)U(kT)$$

$$\text{即: } X(n+1) = \Phi(T)X(n) + \Phi_m(T)U(n) \quad (20-3-6)$$

以上为连续系统离散化后的状态方程差分形式,其中  $X(n+1)$  为本次状态量,  $X(n)$  为上次状态量,若系统状态方程的  $A$ ,  $B$  阵已知后可求得  $\Phi(T)$ ,  $\Phi_m(T)$  从而可求出  $X$  值。

误差修正:

(20-3-6) 式是在采样周期  $T$  内  $U$  保持不变的前提下导出的。实际上  $U$  在两采样时刻之间是变化的。这样就会引起误差。为了减少这部分误差,可假定在两采样时刻之间输入量为一直线函数,即在  $nT$  与  $(n+1)T$  之间有一个  $\Delta U_n(\tau)$  (见图20-3-2)



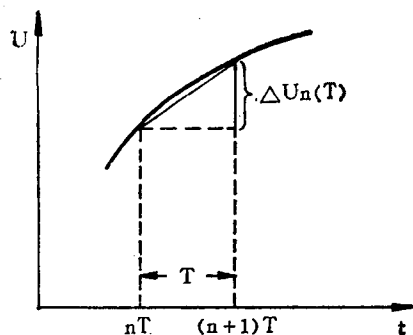


图 20-3-2

$$\Delta U_n(\tau) = \frac{U(n+1) - U(n)}{T} \tau \cong \dot{U}(n) \tau$$

由于  $\Delta U_n(\tau)$  的存在对  $X(n+1)$  引起的变化:

$$\Delta X(n+1) = \int_0^T e^{A(T-\tau)} B \Delta U_n(\tau) d\tau$$

$$= \dot{U}(n) \int_0^T \tau e^{A(T-\tau)} B d\tau$$

$$= \dot{U}(n) \hat{\Phi}_m(T)$$

由此 (20-3-6) 式可以写为:

$$X(n+1) = \Phi(T)X(n) + \Phi_m(T)U(n) + \hat{\Phi}_m(T)\dot{U}(n)$$

输出方程:

$$Y(n+1) = CX(n+1) + DU(n+1)$$

由于计算本次输出量  $Y(n+1)$  时尚不知  $U(n+1)$  故近似用上次输入量  $U(n)$  代入, 为减少误差, 在后面再增加一项  $\dot{U}(n)T$

$$\therefore Y(n+1) = CX(n+1) + D(U(n) + \dot{U}(n)T)$$

(2) 典型环节离散状态方程系数的确定

由以上推导的结论可知, 只要求得系统离散化后的状态方程的系数阵  $\Phi(T)$ 、 $\Phi_m(T)$ 、 $\hat{\Phi}_m(T)$  即可确定系统的各状态变量。而通常一个自动控制系统总是由一些典型环节构成, 如果我们对每一个典型环节事先求得其系数阵, 则可大大方便程序的使用, 目前本仿真程序中包含有以下几种典型环节。

① 积分环节: (图20-3-3)

$$\frac{Y(s)}{U(s)} = \frac{K}{S}$$

$$\dot{X} = KU$$

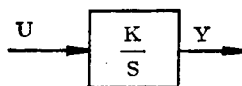


图 20-3-3

② 比例积分环节 (图20-3-4)

$$\frac{Y(s)}{U(s)} = \frac{K_1}{S} + K_2$$

$$\begin{cases} \dot{X} = U \\ Y = K_1 X + K_2 U \end{cases}$$

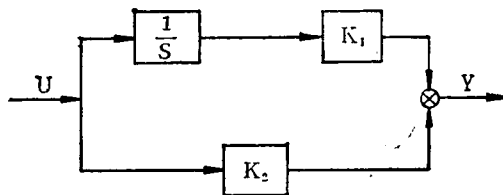


图 20-3-4

③ 惯性环节 (图20-3-5)

$$\frac{Y(s)}{U(s)} = \frac{K}{s+a}$$

$$\begin{cases} \dot{X} = -ax + KU \\ Y = X \end{cases}$$

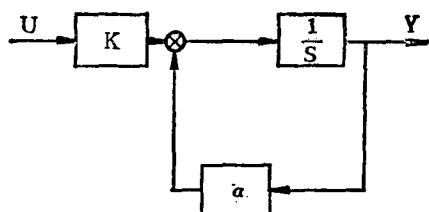


图 20-3-5

④ 比例加惯性环节 (图20-3-6)

$$\frac{Y(s)}{U(s)} = \frac{s+b}{s+a} = 1 + \frac{b-a}{s+a}$$

$$\begin{cases} \dot{X} = -ax + (b-a)u \\ Y = U + X \end{cases}$$

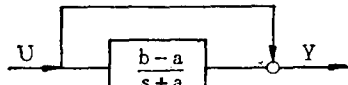


图 20-3-6

⑤ 比例环节 (图20-3-7)

$$X = Y = KU$$

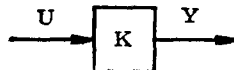


图 20-3-7

各典型环节的 $\Phi(T)$ ,  $\Phi_m(T)$ ,  $\hat{\Phi}_m(T)$ 及状态、差分方程计算公式见附表 (20-3-1)

为了便于使用, 本程序各典型环节均写成统一形式:  $\frac{C_i + D_i S}{A_i + B_i S}$  进入程序后即自动判断属哪一种典型环节, 并计算相应的 X、Y 值。

系统的连接情况仍用连接矩阵 W 来表示, 其意义与连续系统构结图法数字仿真程序(第廿章§2)中的连接矩阵相同。因此各个环节的输入量 U 可表示为:

$$U = WY$$

其中 U 是 n 维列向量, Y 是 n+1 维列向量, W 为 n·(n+1) 阵。

(3) 非线性系统的数字仿真:

本仿真程序由于每算一步各个环节的输入、输出量都要重新算一遍, 因此若在系统中某些环节的入口或出口有非线性环节则很容易处理。在本仿真程序中包含三种常用的非线性环节:

- ① 饱和非线性: (图20-3-8)
- ② 失灵区非线性: (图20-3-9)
- ③ 齿轮间隙非线性: (图20-3-10)

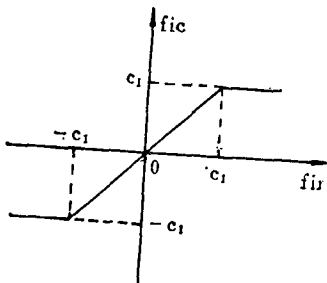


图 20-3-8

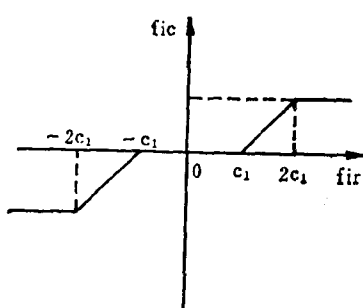


图 20-3-9

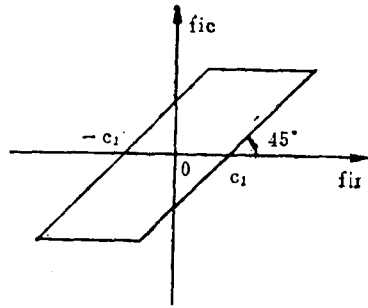


图 20-3-10

以上三种非线性环节已编成相应的子程序, 若某环节入口包含有非线性环节, 只需在计算得到该环节的输入 U 后调用相应的非线性环节子程序, 求得经非线性环节后的输出  $f_{ic}$  作为此环节的输入量。若非线性环节在某环节的出口处, 则只要计算出输出量 Y 后再调用相应的非

线性子程序求得纯非线性环节后的输出量  $f_{ic}$  作为该环节的输出量。在程序中用数组单元

$Z(I)$  标志第  $I$  个环节的非线性情况:

$Z(I)=0$  表示该环节无非线性

$Z(I)=1$  表示该环节前有饱和非线性

$Z(I)=2$  表示该环节前有失灵区非线性

$Z(I)=3$  表示该环节前有齿轮间隙非线性

$Z(I)=4$  表示该环节后有饱和非线性

$Z(I)=5$  表示该环节后有失灵区非线性

$Z(I)=6$  表示该环节后有齿轮间隙非线性

$S(I)$  表示该非线性环节的  $C_1$  值。

## 二、程序说明:

本程序包括以下几部分:

(1) 初始化程序块: 对所需的数组单元定维、置初态, 所有工作单元置初值。

(2) 输入程序块: 输入系统参数如系统环节数, 计算步距, 打印间隔, 打印总数, 系统各环节参数, 非线性类型、参数、连接矩阵参数等。

(3) 运行程序块: 求出各环节的  $\Phi(T)$ ,  $\Phi_m(T)$ ,  $\hat{\Phi}_m(T)$ , 根据连接矩阵求出各环节的输入量  $U_i$ ; 若有入口非线性环节则分别转入相应的非线性子程序; 再由输入量  $U_i$  计算各状态量  $X_i$ ; 最后按输出方程求各环节输出量  $Y_i$ 。若环节出口有非线性则再转相应的非线性子程序处理。计算完一遍后步距增加一步, 重复以上运算, 直至到达要求的打印点数为止。

(4) 输出程序块: 根据使用者要求可以用电传、宽打或 X-Y 记录仪、图像显示仪得到所需的计算结果。

程序使用工作单元:

$N$ : 系统中线性环节数

$R$ : 系统输入阶跃函数幅度

$L_0$ : 计算步距

$L_1$ : 打印间隔 (即每隔  $L_0 \cdot L_1$  打印一点)

$L_2$ : 打印次数

$N_0$ : 打印变量个数

$N_1$ :  
 $N_3$ :  
 $N_4$ :  
} 打印变量编号

$N_5$ : 输出形式标志

( $N_5=0$  电传输出  $N_5=1$  显示图形)

$N_6$ : 绘图变量编号

$N_7$ : 时间比例尺 (X坐标)

$N_8$ : 变量比例尺 (Y坐标)

$A(I)$ :  
 $B(I)$ :  
 $C(I)$ :  
 $D(I)$ :  
} 第  $I$  环节参数

$E(I)$ : 第  $I$  环节参数  $\Phi(T)$   
 $F(I)$ : 第  $I$  环节参数  $\Phi_m(T)$   
 $G(I)$ : 第  $I$  环节参数  $\hat{\Phi}_m(T)$   
 $X(I)$ : 第  $I$  环节状态量  
 $Y(I)$ : 第  $I$  环节输出量  
 $U(I)$ : 第  $I$  环节输入量  
 $W(I)$ : 第  $I$  环节的  $\dot{U}$   
 $Z(I)$ : 第  $I$  环节非线性类型 (详细见本节一.3说明)  
 $S(I)$ : 第  $I$  环节中非线性参数  $C_i$   
 $O(I)$ : 非线性环节上次输入量  
 $Q(I)$ : 非线性环节上次输出量  
 $H(I)$ : 第  $I$  环节类型  
 $P(I, J)$ : 连接矩阵

### 三、操作说明

(1) 画出系统结构图, 各线性环节参数写成  $\frac{C_i + D_i S}{A_i + B_i S}$  并标出环节序号及系统中非线性环节类型。

(2) 写出系统各环节参数

系统中环节参数用数据语句给出, 本程序规定由1000号开始按环节的顺序排列, 如

第一个环节 1000 DATA A(1), B(1), C(1), D(1), X(1), Y(1), Z(1), S(1)

第二个环节 1001 DATA A(2), B(2), C(2), D(2), X(2), Y(2), Z(2), S(2)

(3) 写出系统连接情况, 即写出连接矩阵参数, 写法同本章§2中所述, 本程序规定连接矩阵的数据输入从语句标号1050开始。

(4) 以上准备工作完成后, 可以由BASIC引入本程序, 键盘命令RUN↵, 计算机要求给以下三组变量赋值, 使用者用键盘顺次回答以上变量值后, 计算机自动进入计算。

$N, R, L_0, L_1, L_2$

$N_0, N_2, N_3, N_4, N_5$

$N_6, N_7, N_8$

(5) 计算完毕按要求打印出结果或画出曲线。

### 四、使用注意事项

(1) 本程序是将系统分成若干典型环节, 分别在每个典型环节的输入、输出端加虚拟采样开关和保持器使其离散化, 这样会给计算带来误差, 根据多次计算, 若采样间隔选为  $1/(30 \sim 50)w_c$ , 那么大约可达到0.5%左右精度,  $w_c$  为系统中反应最快的小闭环的开环剪切频率。

(2) 在本程序的环节参数中, 除了比例环节外, 其它环节中不允许  $B_i = 0$ , 即不允许出现纯微分环节。

(3) 本程序只限于单输入系统, 且输入函数为阶跃函数, 若输入非阶跃函数, 则要将程序中  $Y(0)$  按输入信号函数给定, 使用者可按需要自行修改程序相应部分。

(4) 本程序附有三种非线性类型, 使用者若要求有其它非线性可参照以上三种非线性子程序格式自行编制子程序作为补充。

附表 (20-3-1)

典型 环节 H	$\phi(T)$	$\phi_m(T)$	$\hat{\phi}_m(T)$	差 分 方 程
0	1	KT	$\frac{KT^2}{2}$	$X(n+1)\phi=(T)X(n)+\phi_m(T)U(n)$ $+\hat{\phi}_m(T)\dot{U}(n)$ $y(n+1)=X(n+1)$
1	1	KT	$\frac{KT^2}{2}$	$X(n+1)=\phi(T)X(n)+\phi_m(T)U(n)$ $+\hat{\phi}_m(T)\dot{U}(n)$ $y(n+1)=X(n+1)+KbU(n)+KbT\dot{U}(n)$
2	$e^{-aT}$	$\frac{K}{a}(1-e^{-aT})$	$-\frac{K}{a^2}+\frac{K}{a^2}e^{-aT}+\frac{K}{a}T$	$X(n+1)=\phi(T)X(n)+\phi_m(T)U(n)$ $+\hat{\phi}_m(T)\dot{U}(n)$ $y(n+1)=X(n+1)$
3	$e^{-aT}$	$\frac{K}{a}(1-e^{-aT})$	$-\frac{K}{a^2}+\frac{K}{a^2}e^{-aT}+\frac{K}{a}T$	$X(n+1)=\phi(T)X(n)+\phi_m(T)U(n)$ $+\hat{\phi}_m(T)\dot{U}(n)$ $y(n+1)=(b-a)X(n+1)+KU(n)$ $+KT\dot{U}(n)$
4	0	K	0	$X(n+1)=\phi(T)X(n)+\phi_m(T)U(n)$ $+\hat{\phi}_m(T)\dot{U}(n)$ $y(n+1)=X(n+1)$

## 五、试题:

1. 给出四阶随动系统结构图:

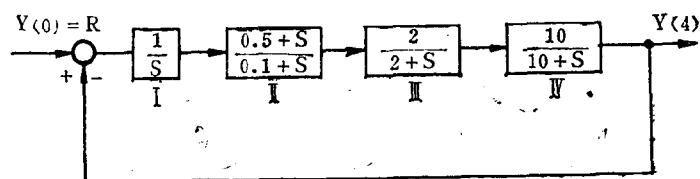


图20-3-11

系统共有4个线性环节, 设 $R=10$ , 各环节的状态变量及输出量的初值均为0。

各环节参数输入语句:

```

1000 DATA 0, 1, 1, 0, 0, 0, 0, 0
1001 DATA 0.1, 1, 5, 1, 0, 0, 0, 0
1002 DATA 2, 1, 2, 0, 0, 0, 0, 0
1003 DATA 10, 1, 10, 0, 0, 0, 0, 0

```

写出系统连接矩阵数据:

```

1050 DATA 1, 0, 1, 1, 4, -1, 2, 1, 1, 3, 2, 1, 4, 3, 1, 0, 0, 0

```

运行结果:

RUN

N, R0, L0, L1, L2

? 4? 10? 0.02? 10? 20

N0, N2, N3, N4, N5

? 1 ? 4 ? 0 ? 0 ? 0

T	Y(4)
0	0
0.2	0.166873
0.4	0.871698
0.6	2.03368
0.8	3.49552
1.0	5.11169
1.2	6.76055
1.4	8.34553
1.6	9.79408
1.8	11.0558
2.0	12.1002
2.2	12.9134
2.4	13.4958
2.6	13.859
2.8	14.0228
3.0	14.0128
3.2	13.8578
3.4	13.5883
3.6	13.2344

END AT 215

2. 给出具有非线性系统结构图:

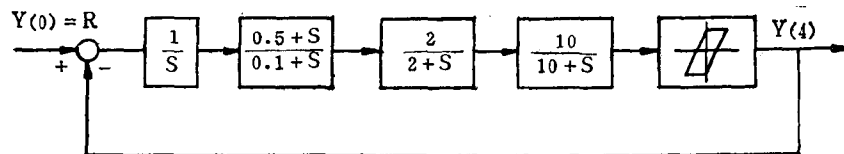


图 20-3-12

系统参数同例 1, 其中环节 IV 输出端具有齿隙非线性。其中环节参数输入语句 1003 修改为:

1003 DATA 10, 1, 10, 0, 0, 0, 6, 1

运行结果:

RUN

N, R<sub>0</sub>, L<sub>0</sub>, L<sub>1</sub>, L<sub>2</sub>

? 4 ? 10 ? 0.02 ? 10 ? 20

N<sub>0</sub>, N<sub>2</sub>, N<sub>3</sub>, N<sub>4</sub>, N<sub>5</sub>

? 1 ? 4 ? 0 ? 0 ? 0

T	Y(4)
0	0
0.2	0
0.4	0
0.6	1.08163
0.8	2.63548
1.0	4.38212
1.2	6.18466
1.4	7.93304
1.6	9.54374
1.8	10.9578
2.0	12.1385
2.2	13.0678
2.4	13.7441
2.6	14.1779
2.8	14.3897
3.0	14.4214
3.2	14.4214
3.4	14.4214
3.6	14.4214

END AT 215

#### 六、程序清单

```

5  PRINT "N,R0,L0,L1,L2"
10  INPUT N,R0,L0,L1,L2
11  PRINT
13  PRINT "N0,N2,N3,N4,N5="
14  INPUT N0,N2,N3,N4,N5
15  PRINT
16  LET L=L0
18  DIM A(N),B(N),E(N),F(N),G(N),X(N)
20  DIM R(L2)
21  DIM C(L2),Z(L2),D(L2),S(L2),U(N),V(N),W(N),H(N),Y(N),K(N)
22  DIM P(N,N+1)
23  DIM O(N)
24  DIM Q(N)
25  FOR I=1 TO N
30    READ A(I),B(I),C(I),D(I),X(I),Y(I),Z(I),S(I)
35  NEXT I
38  FOR I=1 TO N
40    FOR J=0 TO N

```

```

42      LET P(I,J)=0
44      NEXT J
46      NEXT I
48      READ I1,J1,P1
50      IF I1+J1=0 GOTO 55
52      LET P(I1,J1)=P1
54      GOTO 48
55      GOTO 400
56      FOR I=1 TO N
57          LET W(I)=0
58          IF Z(I)=6 GOTO 63
60          LET Q(I)=0
61          LET O(I)=0
62          GOTO 65
63          LET Q(I)=Y(I)
64          GOTO 61
65          IF H(I)=0 GOTO 90
68          IF H(I)=1 GOTO 90
69          IF H(I)=4 GOTO 100
70          LET E(I)=EXP(-A(I)*L)
75          LET F(I)=K(I)*(1-E(I))/A(I)
80          LET G(I)=K(I)*(E(I)-1)/(A(I)*A(I))+K(I)*L/A(I)
85      NEXT I
86      GOTO 98
90      LET E(I)=1
92      LET F(I)=K(I)*L
94      LET G(I)=K(I)*L*L/2
96      NEXT I
98      LET T=0
99      GOTO 120
100     LET E(I)=0
101     LET F(I)=K(I)
102     LET G(I)=0
103     NEXT I
104     GOTO 98
120     GOTO 450
132     LET Y(0)=R0
135     FOR Q1=1 TO L2
140         FOR Q2=1 TO L1
145             GOSUB 300

```



```

348      GOTO 500
150      FOR I=1 TO N
155          LET V(I)=(U(I)-W(I))/L0
160          LET W(I)=U(I)
165          LET X(I)=E(I)*X(I)+F(I)*U(I)+G(I)*V(I)
167          IF H(I)=3 GOTO 173
169          IF H(I)=1 GOTO 176
170          LET Y(I)=X(I)
171      NEXT I
172      GOTO 182
173      LET Y(I)=(B(I)-A(I))*X(I)+K(I)*U(I)+K(I)*L*V(I)
174      GOTO 171
176      LET Y(I)=X(I)+B(I)*K(I)*U(I)+B(I)*K(I)*L*V(I)
178      GOTO 171
182      LET T=T+L0
183      GOTO 800
185      NEXT Q2
205      GOTO 485
210      NEXT Q1
215      END
300      FOR I=1 TO N
305          LET U(I)=0
310          FOR J=0 TO N
315              LET U(I)=U(I)+P(I,J)*Y(J)
320          NEXT J
325      NEXT I
330      RETURN
400      FOR I=1 TO N
401          IF A(I)=0 GOTO 425
402          IF D(I)=0 GOTO 410
403          LET H(I)=3
404          LET K(I)=D(I)/B(I)
405          LET A(I)=A(I)/B(I)
406          LET B(I)=C(I)/D(I)
409          GOTO 445
410          IF B(I)=0 GOTO 416
411          LET H(I)=2
412          LET K(I)=C(I)/B(I)
413          LET A(I)=A(I)/B(I)
414          LET B(I)=0

```

```

415     GOTO 445
416     LET H(I)=4
417     LET K(I)=C(I)/A(I)
418     LET A(I)=0
419     LET B(I)=0
420     GOTO 445
425     IF D(I)=0 GOTO 435
426     LET H(I)=1
427     LET K(I)=C(I)/B(I)
428     LET A(I)=0
429     LET B(I)=D(I)/C(I)
430     GOTO 445
435     LET H(I)=0
436     LET K(I)=C(I)/B(I)
437     LET A(I)=0
438     LET B(I)=0
445     NEXT I
446     GOTO 56
450     IF N5=1 GOTO 1500
455     IF N0=3 GOTO 479
456     IF N0=2 GOTO 470
460     PRINT "T",TAB(10),"Y(",N2,")"
461     PRINT
462     PRINT T,TAB(10);Y(N2)
463     GOTO 484
470     PRINT "T",TAB(10),"Y'(",N2,")",TAB(20),"Y(",N3,")"
471     PRINT
472     PRINT T,TAB(10);Y(N2);TAB(20);Y(N3)
473     GOTO 484
479     PRINT "T",TAB(10),"Y(",N2,")",
480     PRINT TAB(20);"Y(",N3,")",TAB(30);"Y(",N4,")"
482     PRINT T,TAB(10);Y(N2);TAB(20);Y(N3);TAB(30);Y(N4)
484     GOTO 132
485     IF N5=1 GOTO 1515
486     IF N0=3 GOTO 492
487     IF N0=2 GOTO 490
488     PRINT T, TAB(10);Y(N2); TAB(20+2*Y(N2)),"."
489     GOTO 493
490     PRINT T, TAB(10);Y(N2); TAB(20);Y(N3)
491     GOTO 493

```

```

492 PRINT T; TAB(10);Y(N2); TAB(20);Y(N3); TAB(30);Y(N4)
493 GOTO 210
500 FOR I=1 TO N
505     IF Z(I)=0 GOTO 555
510     IF Z(I)=1 GOTO 520
515     IF Z(I)=2 GOTO 540
516     IF Z(I)=3 GOTO 551
518 NEXT I
519 GOTO 150
520 LET M=I
525 LET C1=S(I)
530 GOTO 210
535 GOTO 555
540 LET M=I
545 LET C1=S(I)
548 GOSUB 650
550 GOTO 555
551 LET M=I
552 LET C1=S(I)
553 GOSUB 900
555 NEXT I
560 GOTO 150
600 IF ABS (U(M))>=C1 GOTO 615
605 LET U(M)=U(M)
610 GOTO 635
615 IF U(M)>=C1 GOTO 630
620 LET U(M)=-C1
625 GOTO 635
630 LET U(M)=C1
635 RETURN
650 IF ABS (U(M))>=C1 GOTO 665
655 LET U(M)=0
660 GOTO 685
665 IF U(M)>=C1 GOTO 680
670 LET U(M)=U(M)+C1
675 GOTO 685
680 LET U(M)=U(M)-C1
685 RETURN
800 FOR I=1 TO N
801     IF Z(I)=4 GOTO 810

```

```

802     IF Z(I)=5 GOTO 820
803     IF Z(I)=6 GOTO 840
808     NEXT I
809     GOTO 185
810     LET U(I)=Y(I)
812     LET C1=S(I)
814     LET M=I
816     GOSUB 600
818     LET Y(I)=U(I)
819     GOTO 808
820     LET U(I)=Y(I)
821     LET M=I
822     LET C1=S(I)
823     GOSUB 650
824     LET Y(I)=U(I)
825     GOTO 808
840     LET U(I)=Y(I)
841     LET M=I
842     LET C1=S(I)
843     GOSUB 900
844     LET Y(I)=U(I)
845     GOTO 808
900     IF U(M)>O(M) GOTO 904
902     GOTO 916
904     LET O(M)=U(M)
905     IF Q(M)<=(U(M)-C1) GOTO 912
906     LET U(M)=Q(M)
908     LET Q(M)=U(M)
910     RETURN
912     LET U(M)=U(M)-C1
914     GOTO 908
916     IF U(M)<O(M) GOTO 920
918     LET O(M)=U(M)
919     GOTO 908
920     IF Q(M)>=(U(M)+C1) GOTO 924
922     LET O(M)=U(M)
923     GOTO 908
924     LET O(M)=U(M)
926     LET U(M)=U(M)+C1
927     GOTO 908

```

```

1000 DATA 1, 10, 5, 10, 0, 0, 0, 0
1001 DATA 1, 5, 1, 0, 0, 0, 0, 0
1002 DATA 1, 1, 1, 0, 0, 0, 0, 0
1003 DATA 0, 1, 1, 0, 0, 0, 0, 0
1050 DATA 1, 0, 1, 1, 4, -1, 2, 1, 1, 3, 2, 1, 4, 3, 1, 0, 0, 0
1500 PRINT "N6,N7,N8"
1501 INPUT N6,N7,N8
1502 PRINT
1504 LET Y2=0
1505 LET Y4=0
1507 PRINT "Y("N6:")=F(T), T(I+1)-T(I)="L1
1510 GOTO 132
1515 LET Y3=(Y(N6)-Y2)*N8
1516 LET Y1=INT (Y3+Y4)
1517 LET Y2=Y(N6)
1518 LET Y4=Y3-Y1+Y4
1525 PLOT 1, N7,Y1 (CRT显示语句, 为DJS-131机所特有语句)
1530 GOTO 210

```

## §4 单纯形法寻优程序

### 一、方法概要

本程序可用来对N个变量的指标函数进行寻优计算。

在闭环控制系统中, 输入阶跃函数R, 给定寻优参数初始值, 计算闭环系统的动态响应, 误差信号和指标函数。

当指标函数满足给定寻优精度E。后, 认为寻优结束, 找到了最优的参数, 当指标函数不满足给定精度E。时, 用单纯形法改变寻优参数, 重新计算系统动态响应, 误差信号和指标函数直至满足精度为止。

(1) 单纯形法寻优: 单纯形法是多变量参数寻优的一种方法, 此方法可不用计算函数的导数值, 而是采用先算出若干个点的函数值, 然后对它们进行比较, 从它们之间的大小关系, 就可看出函数变化的大致趋势。

如下图所示, 对于两个变量:  $\alpha_1$ 、 $\alpha_2$  图中画出 $\theta(\alpha)=C$ 的等高线族, 若先计算1、2、3三点(它们构成一个三角形, 边长由初始步长 $L_0$ 决定)的函数值, 然后对它们的大小进行比较, 其中:

$$\theta_1 > \theta_2 > \theta_3$$

$\theta_1$ 为最坏点,  $\theta_2$ 为次坏点,  $\theta_3$ 为最好点。故将1点弃去, 取1点的反射点4点为新的点, 再比较2、3、4点 $\theta$ 值的大小, 其中:

$$\theta_2 > \theta_3 > \theta_4$$

$\theta_2$ 为最坏点,  $\theta_3$ 为次坏点,  $\theta_4$ 为最好点。故将2点弃去, 再取2点的反射点5点, 比较3、4、5三点的 $\theta$ 值, ……如此循环下去, 最后可找到最小点的 $\theta$ 值。

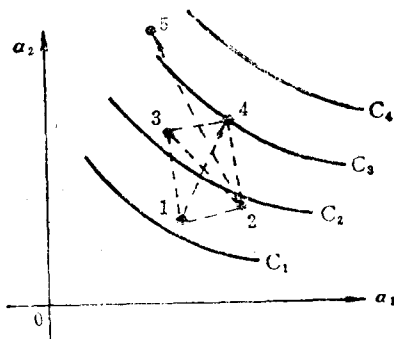


图 20-4-1

在本程序中求出反射点后，计算反射点的 $\theta$ 值，若此 $\theta$ 值还小于次坏点的 $\theta$ 值，则可进行扩张，扩张系数为 $\mu$ ，且进一步验算  $(1-\mu)\theta_{\text{最坏}} + \mu\theta_{\text{反射}} < \theta_{\text{最好}}$ 。若满足，则进行扩张，扩张点值  $X = (1-\mu)X_{\text{最坏}} + \mu X_{\text{反射}}$ 。若不满足，则只以反射点取代最坏点。

若计算出的反射点不小于次坏点的 $\theta$ 值，但小于最坏点 $\theta$ 值，则进行压缩，压缩系数为 $\lambda$ ，压缩点值  $X = (1-\lambda)X_{\text{最坏}} + \lambda X_{\text{反射}}$ ，算出 $\theta$ 值，此 $\theta$ 值若小于 $\theta_{\text{次坏}}$ ，则以此点取代最坏点。若计算出的反射点既不小于次坏点的 $\theta$ 值，也不小于最坏点的 $\theta$ 值，则仍用最坏点来进行压缩，压缩后的 $\theta$ 值如小于次坏点 $\theta$ 值，则用此点取代最坏点。

以上几种情况，在电传机上均打出一个“-”号。

若压缩后的点，其 $\theta$ 值不满足小于 $\theta_{\text{次坏}}$ ，则整个单纯形进行收缩，这时电传机上打印出一个“+”值。

(2) 利用计算机实现 P、I、D 调节器：在本程序中是对 P、I、D 参数进行寻优，因此存在如何用计算机实现 P、I、D 调节器的问题。

P、I、D 调节器输入和输出的关系式如下：

$$U(t) = K_p \left[ e + \frac{1}{T_i} \int_0^t e dt + T_D \frac{de}{dt} \right]$$

其中：

$K_p$ ——比例系数。

$T_i$ ——积分时间常数。

$T_D$ ——微分时间常数。

写成离散形式，采样周期为  $T$ ，初始值为 0：

$$U(n) = K_p \left[ e(n) + \frac{1}{T_i} \sum_{k=0}^n e(k)T + T_D \frac{e(n) - e(n-1)}{T} \right]$$

变成递推的形式：

$$U(n) - U(n-1) = K_p \left\{ e(n) - e(n-1) + \frac{T}{T_i} e(n) + \frac{T_D}{T} (e(n) - 2e(n-1) + e(n-2)) \right\}$$

$$U(n) = U(n-1) + K_p \left\{ e(n) - e(n-1) + \frac{T}{T_i} e(n) + \frac{T_D}{T} [e(n) - 2e(n-1) + e(n-2)] \right\}$$

其中：

$U(n)$ ——本次输出量

$U(n-1)$ ——前次输出量  
 $e(n)$ ——本次输入量  
 $e(n-1)$ ——前次输入量  
 $e(n-2)$ ——再前一次输入量

(3) 纯时间延滞环节的仿真：在本程序的控制对象中有一大时间延滞环节，对于这样的环节我们是利用内存单元移位实现延时的作用，即将时间延时环节的输入量送至内存  $(A+1)$  单元， $A = \frac{T}{L_1}$ ， $T$  为延时环节的延时时间， $L_1$  为计算步距。由内存  $(1)$  单元作为延时环节的输出，靠平移使存入的量一拍拍移送至  $(1)$  单元，如下图所示：

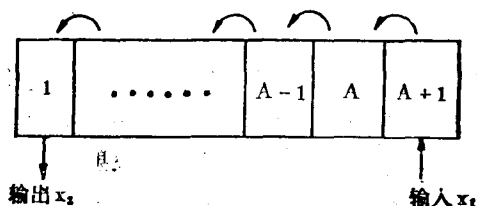


图 20-4-2

其程序如下：

```

10 FOR I=1 TO L1
15 LET E(A+1)=X1      存入
20 LET X2=E(1)          取出
25 FOR I3=1 TO A+1
30 LET E(I3-1)=E(I3)    } 平移
35 NEXT I3
    
```

(4) 动态响应的计算：本程序的动态响应计算采用离散相似法。

(5) 指标函数的计算：指标函数  $\theta$  采用下式计算：

$$Q = \int_0^{\infty} |e| t \Delta t = \sum_{i=1}^T |e| T L_1$$

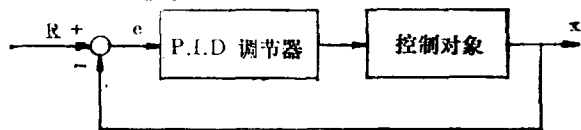


图 20-4-3

其中  $e$  为误差信号

$T$  为采样周期

$L_1$  为计算步距

## 二、程序说明：

(1) 由于动态响应的计算采用离散相似法进行，因此离散相似法使用的限制对它也适用。

(2) 调节器  $P$ 、 $I$ 、 $D$  初值的确定：根据控制对象的参数，来确定  $P$ 、 $I$ 、 $D$  的初始值，对于程序中的系统是两个一阶惯性环节、一个延时环节，如图 (20-4-4) 所示：要使系统输出仅仅具有滞后  $T_s$ ，其他均能不失真地反映给定值  $R(s)$ ，即  $X_1(y) = R(t - T_s)$  可推导出  $P$ 、 $I$ 、 $D$  调节器参数如下：

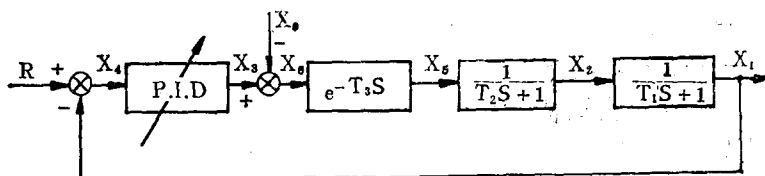


图 20-4-4

$$K_p = \frac{T_1 + T_2}{KT_s}$$

$$T_i = T_1 + T_2$$

$$T_D = \frac{T_1 T_2}{T_1 + T_2}$$

由于是按连续系统推导出的 $K_p$ 、 $T_i$ 、 $T_D$ 值，因此用在离散化的控制系统中 $K_p$ 值会偏大，故选用 $K_p$ 初始值时要适当缩小此计算值。

(3) 程序中使用的变量：

X数组——被寻优的参数变量

A数组——被寻优的参数变量上限

B数组——被寻优的参数变量下限

Q数组——指标函数，其中

$Q_0$ ——最坏点（最大）的指标函数

$Q_1$ ——次坏点的指标函数

$Q_2$ ——最好点的指标函数

E数组——产生时间延时环节的数组

N——被寻优的参数变量的个数

$E_0 * Q_0$ ——当单纯形中最大的指标函数与最小的指标函数之差小于 $E_0 * Q_0$ 时，认为最优找到。

$$Q_0 - Q_2 < E_0 * Q_0$$

$M_0$ ——允许的最大搜索次数

$L_0$ ——单纯形初始步长

$L_1$ ——控制对象的计算步距

$L_2$ —— $L_1 * L_2 = T$ ，T为系统的采样周期

$L_3$ ——保证系统到达稳态，需要计算的总次数。

$X_1$ ——控制对象的输出

$X_2$ ——第一个惯性环节的输出

$X_3$ ——时间延时环节的输出

$X_4$ ——P、I、D调节器的输出

$X_5$ ——P、I、D调节器的输入，即系统输入信号R与输出信号 $x_1$ 之差值，也就是误差信号e。

$X_6$ ——外界扰动

$X_7$ ——P、I、D输出 $X_4$ 与扰动值 $X_6$ 之差



$W_2$ ——前一次 $X_2$ 值  
 $V_2$ —— $\dot{X}_2$ 值  
 $W_1$ ——前一次 $X_1$ 值  
 $U_1$ ——前两次 $X_1$ 值

### 三、操作说明:

(1) 300号子程序是用来计算指标函数的子程序, 现控制对象是两个一阶惯性环节和一个延时环节, 调节器采用P、I、D调节器, 若控制对象不同, 调节器改变, 须重新编300号子程序。

(2) 250号子程序是用来判断参数变量是否超出上、下限, 若越限, 则选择上、下限值给变量赋值。

(3) 400号~418号程序是读入控制对象参数及计算环节的系数值, 若控制对象不同, 须重新编写该程序。

(4) 500号程序为输入控制对象参数的数值程序, 501号~503号程序为给定寻优变量的初始值, 上限, 下限。

在程序中控制对象给定参数采用标么值计算即: 对于上图中的系统

$$\text{取 } T_z = T_1 + T_2 + T_3$$

$T_1, T_2, T_3$  为系统给定值

$$T_1 = \frac{T_1}{T_z} \quad \text{为 } T_1 \text{ 的标么值}$$

$$T_2 = \frac{T_2}{T_z} \quad \text{为 } T_2 \text{ 的标么值}$$

$$T_3 = 1 - T_1 - T_2$$

且取

$$A_1 = \frac{T_1}{T_2}$$

因此程序中只需输入 $T_z, A_1$ 值即可,  $T_1, T_2$ 值即可计算出。在图中标注的均为标么值。

(5) 本程序当寻优参数选好后, 由宽行打印机打出整个系统的动态响应值及过渡过程曲线。且当过渡过程稳定后, 在扰动处加 $X_0 = -10$ 的扰动量, 并由宽打打印出输出值及过渡过程曲线。

### 四、试题:

具有纯延时环节及两个一阶惯性环节的系统, 采用P、I、D调节器。寻P、I、D参数的最优值。

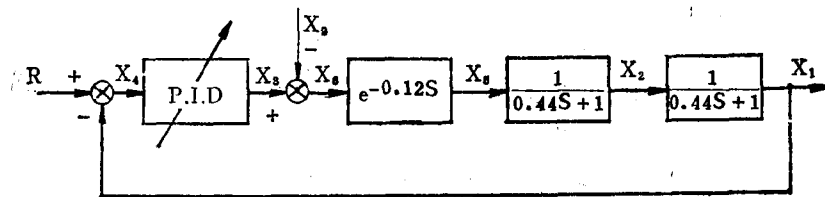


图 20-4-5

#### (1) 输入控制系统参数

500 DATA 0.44, 1, 0.02, 10, 30.

0.44为T,值, i 为A,值,

0.02为L<sub>1</sub>值

10为L<sub>2</sub>值

30为L<sub>3</sub>值

(2) 输入P、I、D、初始值及上限值、下限值

501 DATA 1.5, 2.5, 0.5 (为比例环节数值)

502 DATA 0.8, 1, 0.5 (为积分环节数值)

503 DATA 0.2, 0.5, 0.1 (为微分环节数值)

(3) 回答电传问话:

N, E<sub>0</sub>, M<sub>0</sub>, L<sub>0</sub>

3 0.01 50 0.05

(寻优变量 N=3, 寻优精度 E<sub>0</sub>=0.01, 寻优最大次数 M<sub>0</sub>=50, 单纯形初始步长 L<sub>0</sub>=0.05)

## 五、程序清单及运行结果

### 程序清单

```
10 PRINT "N, E0, M0, L0"
12 INPUT N, E0, M0, L0
14 PRINT
15 GOTO 400
16 DIM X(N, N+3), Q(N+3)
17 DIM A(N), B(N)
18 LET I0=0
20 FOR I=1 TO N
22   READ X(I,0)
23   READ A(I), B(I)
24 NEXT I
25 LET M1=0
26 FOR J=1 TO N
28   FOR I=1 TO N
30     IF I=J GOTO 36
32     LET X(I,J)=X(I,0)
34     GOTO 38
36     LET X(I,J)=X(I,0)+L0
38   NEXT I
39 NEXT J
40 FOR J=0 TO N
41   GOSUB 250
42   GOSUB 300
44   LET Q(J)=Q1
46 NEXT J
```

```

50 LET Q0=0
52 FOR J=0 TO N
54     IF Q(J)>Q0 GOTO 58
56     GOTO 62
58     LET Q0=Q(J)
60     LET J0=J
62 NEXT J
64 LET Q1=0
66 FOR J=0 TO N
68     IF J=J0 GOTO 78
70     IF Q(J)>Q1 GOTO 74
72     GOTO 78
74     LET Q1=Q(J)
76     LET J1=J
78 NEXT J
80 LET Q2=Q0
82 FOR J=0 TO N
84     IF Q(J)<Q2 GOTO 88
86     GOTO 92
88     LET Q2=Q(J)
90     LET J2=J
92 NEXT J
94 LET M1=M1+1
96 IF(Q0-Q2)<(E0*Q0) GOTO 220
100 IF M1>M0 GOTO 240
102 FOR I=1 TO N
104     LET X(I, N+1)=0
106     FOR J=0 TO N
108         LET X(I, N+1)=X(I, N+1)+X(I, J)
110     NEXT J
112     LET X(I, N+1)=2*(X(I, N+1)-X(I, J0))/(N-X(I, J0))
114 NEXT I
116 LET J=N+1
117 GOSUB 250
118 GOSUB 300
120 LET Q(N+1)=Q1
122 IF Q(N+1)<Q(J1) GOTO 174
124 IF Q(N+1)<Q(J0) GOTO 138
126 FOR I=1 TO N
128     LET X(I, N+2)=X(I, N+1)

```

```

130     LET X(1, N+1)=X(1, J0)
132     LET X(1, J0)=X(1, N+2)
134     NEXT I
138     FOR I=1 TO N
140     LET X(1, N+2)=.3 * X(1, J0)+.7 * X(1, N+1)
142     NEXT I
144     LET J=N+2
146     GOSUB 250
148     GOSUB 300
148     LET Q(N+2) =Q1
150     IF Q(N+2)<Q(J1) GOTO 164
152     FOR J=0 TO N
154         FOR I=1 TO N
156             LET X(1, J)=(X(1, J)+X(1, J2))/2
157         NEXT I
158     NEXT J
161     PRINT "+",
162     GOTO 40
164     FOR I=1 TO N
166         LET X(1, J0)=X(1, N+2)
168     NEXT I
170     LET Q(J0)=Q(N+2)
171     PRINT "-",
172     GOTO 50
174     IF (1.5 * Q(N+1) - .5 * Q(J0)) < Q(J2) GOTO 184
176     FOR I=1 TO N
178         LET X(1, N+2)=X(1, N+1)
180     NEXT I
181     LET Q(N+2)=Q(N+1)
182     GOTO 164
184     FOR I=1 TO N
186         LET X(1, N+3)=1.5 * X(1, N+1) - .5 * X(1, J0)
188     NEXT I
190     LET J=N+3
191     GOSUB 250
192     GOSUB 300
194     LET Q(N+3)=Q1
196     IF Q(N+3)<Q(N+1) GOTO 200
198     GOTO 176
200     FOR I=1 TO N

```

```

202     LET X(I,N+2)=X(I,N+3)
204 NEXT I
205 LET Q(N+2)=Q(N+3)
206 GOTO 164
220 PRINT "QPTIM X(I)"
222 FOR I=1 TO N
224     PRINT "X", I,X(I,J2)
226 NEXT I
228 PRINT "COST FUNCTION"
230 PRINT "Q="; Q(J2)
233 PRINT "T", TAB(10),"X1",TAB(20),"X4",TAB(30),"Q1"
235 LET I9=1
236 LET J=J2
237 GOTO 300
240 PRINT "EXCESS"
242 GOTO 222
250 FOR I=1 TO N
252     IF X(I, J)>A(I) GOTO 258
253     IF X(I, J)<B(I) GOTO 260
254 NEXT I
255 RETURN
258 LET X(I,J)=A(I)
259 GOTO 254
260 LET X(I,J)=B(I)
261 GOTO 254
300 LET K1=X(1,J)
302 LET T4=X(2,J)
304 LET T5=X(3,J)
305 LET W2=0
306 LET X1=0
308 LET X2=0
309 LET X3=0
310 LET W4=0
311 [ FOR I3=1 TO A2+1
312     LET E (I3)=0
313 NEXT I3
314 LET U4=0
315 LET I8=0
316 LET T=0
317 LET X9=0

```

```

318 LET Q1=0
319 LET T6=L1*L2
320 FOR I1=1 TO L3
326 LET X4=10-X1
328 LET X3=X3+K1*(X4-W4+T6*X4/T4+T5*(X4-2*W4+U4)/T6)
329 LET X6=X3-X9
330 LET U4=W4
332 LET W4=X4
333 FOR I2=1 TO L2
334 LET E(A2+1)=X6
336 LET X5=E(1)
338 FOR I3=1 TO (A2+1)
340 LET E(I3-1)=E(I3)
342 NEXT I3
346 LET X2=E2*X2+F2*X5
348 LET V2=(X2-W2)/L1
349 LET T=T+L1
350 LET X1=E1*X1+F1*W2+G1*V2
352 LET W2=X2
354 LET X4=10-X1
355 LET Q1=Q1+ABS(X4)*L1*T
356 NEXT I2
358 IF I9=0 GOTO 366
359 PRINT T, TAB(10), X1, TAB(20), X4, TAB(30), Q1, TAB(40+1.5*X1),
366 NEXT I1
367 IF I9=1 GOTO 370
368 PRINT "K1=", K1, "T4=", T4, "T5=", T5, "Q ", Q1
369 RETURN
370 IF I8=1 GOTO 374
371 LET X9=10
372 LET I8=1
373 GOTO 320
374 PRINT "T1,T2,T3,K1,T4,T5"
375 PRINT T1,T2,T3,K1,T4,T5
376 STOP
400 READ T2,A1,L1,L2,L3
405 LET T1=A1*T2
407 LET T3=1-T2-T1
408 LET A2=INT(T3/L1+.5)
410 LET E1=EXP(-L1/T1)

```

```

412 LET F1=1-E1
413 LET G1=(E1-1)*T1+L1
414 LET E2=EXP(-L1/T2)
416 LET F2=1-E2
418 LET G2=(E2-1)*T2+L1
420 DIM E(A2+1)
430 GOTO 16
500 DATA .44, 1, .02, 10, 30
501 DATA 1.5, 2.5, 1
502 DATA .8, 1, .5
503 DATA .2, .5, .1

```

### 运行结果

RUN

N, E0, M0, L0

? 3 ? .01? 50? .05

K1=1.5	T4=.8	T5 .2	Q1 1.97882
K1=1.55	T4=.8	T5 .2	Q1 2.04066
K1=1.5	T4=.85	T5 .2	Q1 2.26663
K1=1.5	T4=.8	T5 .25	Q1 1.84433
K1=1.53333	T4=.75	T5 .233333	Q1 1.86497
K1=1.55	T4=.7	T5 .25	Q1 1.98795
-K1=1.47222	T4=.766666	T5 .255555	Q1 1.84894
K1=1.43333	T4=.749999	T5 .283333	Q1 2.01657
-K1=1.5037	T4=.744443	T5 .292592	Q1 2.13623
K1= 1.50111	T4=.783333	T5 .227778	Q1 1.79812
-K1=1.44889	T4=.816665	T5 .255555	Q1 1.88133
K1= 1.508	T4=.769999	T5 .24	Q1 1.82152
-K1=1.53385	T4=.802222	T5 .222963	Q1 1.8736
K1= 1.49071	T4=.777332	T5 .245777	Q1 1.81541
-K1=1.49988	T4=.753776	T5 .225703	Q1 1.80918
K1= 1.49982	T4=.730664	T5 .213555	Q1 1.86993
-K1=1.48647	T4=.772961	T5 .226172	Q1 1.77657
K1= 1.4757	T4=.774442	T5 .219259	Q1 1.77524
-K1=1.49375	T4= .7637	T5 .202715	Q1 1.89571
K1= 1.49162	T4=.773243	T5 .232859	Q1 1.78472
-K1=1.47907	T4=.800235	T5 .22756	Q1 1.79573
-K1=1.46315	T4=.781946	T5 .22534	Q1 1.75266
K1= 1.44417	T4=.781253	T5 224122	Q1 1.73468
-K1=1.46192	T4=.752389	T5 .223266	Q1 1.76151

-K1=1.42957	T4=.765479	T5 .211572	Q1 1.74337
K1= 1.39855	T4=.761598	T5 .200928	Q1 1.78163
-K1=1.41474	T4=.758306	T5 .220048	Q1 1.70895
K1=1.38426	T4=.750238	T5 .220442	Q1 1.68717
-K1=1.37675	T4=.778924	T5 .214158	Q1 1.71496
-K1=1.37388	T4=.774796	T5 .227576	Q1 1.69988
K1= 1.34604	T4=.779455	T5 .235578	Q1 1.7508
-K1=1.31243	T4=.754719	T5 .217329	Q1 1.66911
K1= 1.24655	T4=.741452	T5 .213932	Q1 1.68744
-K1=1.33696	T4=.740911	T5 .229406	Q1 1.67889
K1= 1.31707	T4=.721905	T5 .237031	Q1 1.70634
-K1=1.31522	T4=.722448	T5 .217209	Q1 1.67283
K1= 1.28588	T4=.696274	T5 .212025	Q1 1.79563
-K1=1.25881	T4=.728481	T5 .222187	Q1 1.67973
K1=1.29644	T4=.735008	T5 .221664	Q1 1.66188
-K1=1.27909	T4=.733872	T5 .208061	Q1 1.67096

-OPTIM X(I)

X1 1.29644

X2 .735008

X3 .221664

COST FUNCTION

Q=1.66188

T	X1	X4	Q1
.2	.451473	9.54853	.216784
.4	3.95502	6.04498	.68622
.599999	7.44521	2.55479	1.0867
.799999	9.50637	.493633	1.26461
.999999	10.2246	-.224575	1.29404
1.2	10.2096	-.209553	1.3496
1.4	10.0003	-3.47137E-4	1.3741
1.6	9.86611	.133888	1.39966
1.8	9.85509	.144913	1.45037
2	9.91491	8.50906E-2	1.49408
2.2	9.98391	1.60923E-2	1.51333
2.4	10.0289	-2.89173E-2	1.51993
2.6	10.0453	-4.52976E-2	1.54002
2.8	10.0428	-4.28276E-2	1.56437
3	10.0331	-.033102	1.58628
3.2	10.0235	-2.35081E-2	1.60345
3.4	10.0167	-1.67294E-2	1.61637



3.6	10.0126	-1.25847E-2	1.62637	.
3.8	10.01	-9.95827E-3	1.63457	.
4	10.0079	-7.94411E-3	1.64146	.
4.19999	10.0062	-6.16264E-3	1.64716	.
4.39999	10.0045	-4.54712E-3	1.65169	.
4.59998	10.0032	-3.17955E-3	1.65509	.
4.79998	10.0021	-2.13242E-3	1.65751	.
4.99997	10.0014	-1.39046E-3	1.65918	.
5.19997	10.0009	-8.81195E-4	1.66029	.
5.39996	10.0005	-5.43594E-4	1.66102	.
5.59996	10.0003	-3.20435E-4	1.66147	.
5.79995	10.0002	-1.77383E-4	1.66174	.
5.99995	10.0001	-8.7738E-5	1.66188	.
6.19994	9.85373	.146267	1.69652	.
6.39994	8.86702	1.33298	2.62552	.
6.59993	7.13592	2.86408	5.47872	.
6.79993	6.08256	3.91744	10.1775	.
6.99992	5.80719	4.19281	15.8843	.
7.19992	6.16313	3.83687	21.621	.
7.39991	6.83523	3.16477	26.7066	.
7.59991	7.55343	2.44657	30.8583	.
7.7999	8.17105	1.82895	34.0881	.
7.9999	8.64829	1.35171	36.5468	.
8.19988	9.00264	.997362	38.4077	.
8.39987	9.26705	.73295	39.8124	.
8.59985	9.46864	.531363	40.8629	.
8.79984	9.62403	.375969	41.6332	.
8.99982	9.74251	.257486	42.1817	.
9.19981	9.83021	.169786	42.5584	.
9.3998	9.89265	.107347	42.8069	.
9.59978	9.93534	6.46553E-2	42.9636	.
9.79977	9.96347	3.65314E-2	43.057	.
9.99975	9.98138	1.86176E-2	43.1085	.
10.1997	9.99243	7.57217E-3	43.1328	.
10.3997	9.99896	1.04332E-3	43.1404	.
10.5997	10.0026	-2.55775E-3	43.143	.
10.7997	10.0043	-4.28581E-3	43.1507	.
10.9997	10.0048	-4.84276E-3	43.1609	.
11.1997	10.0047	-4.70734E-3	43.1715	.
11.3997	10.0042	-4.21143E-3	43.1816	.

11.5996	10.0036	-3.57437E-3	43.1904	.
11.7996	10.0029	-2.91443E-3	43.1979	.
11.9996	10.0023	-2.30217E-3	43.204	.
T1, T2, T3, K1, T4, T5				
.44	.44	.12	1.29644	.735008
.221664				

## §5 线性定常系统最佳控制仿真设计程序

### 一、方法概要

本程序可用于多输入、多输出的线性定常系统在二次型性能指标下(取终端时间  $t \rightarrow \infty$ ) 实现全状态反馈的最佳控制的设计。当已知系统的状态方程后, 即可使用本程序求得最佳反馈控制系数阵及系统对阶跃信号的反应。

由现代控制理论知道, 对一线性定常系统, 其状态方程可写成如下形式:

$$\dot{X} = AX + BU \quad (20-5-1)$$

$X$  称为状态变量,  $U$  称为控制变量,  $A$ 、 $B$  称为系数矩阵。

系统的结构图如下:

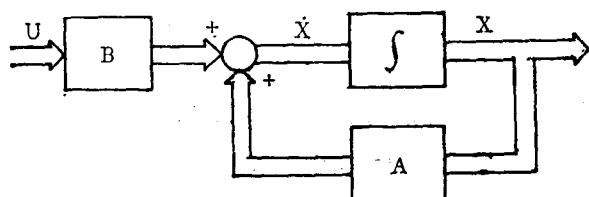


图 20-5-1

要实现对该系统最佳控制, 不同的性能指标下有不同的控制律。

若取终端时间  $t \rightarrow \infty$ , 在二次型性能指标下, 取目标函数为:

$$J = \frac{1}{2} \int_0^{\infty} (X^T Q X + U^T R U) dt \quad (20-5-2)$$

使  $J$  达到最小, 即  $J = J_{min}$  的控制律称为最佳控制律, 此时满足:

$$U = -R^{-1} B^T P X \quad (20-5-3)$$

即为全状态线性反馈。其中  $Q$ 、 $R$  称为加权阵,  $Q$  为半正定阵,  $R$  为正定阵, 而  $P$  为 Riccate 方程的解, 此时 Riccate 微分方程蜕化为代数方程, 为如下形式:

$$PA + A^T P + P B R^{-1} B^T P + Q = 0 \quad (20-5-4)$$

前述各式中角标  $T$  表示矩阵的转置。

解得的  $P$  为非负定对称阵。

因此, 为了确定在前述性能指标下系统的最佳控制律, 需要求解 Riccate 方程。称

$$K = -R^{-1} B^T P$$

为全状态反馈系数阵, 求得  $K$  即完成了对该系统的最佳设计。因此, 求解 Riccate 方程是设计这类系统的关键。

下面对本程序求解Riccate方程的方法加以说明。

将(20-5-4)式两边同加以 $-PBR^{-1}P$ , 并稍加整理后即得:

$$P(A - BR^{-1}B^*P) + (A - BR^{-1}B^*P)^*P = -PBR^{-1}B^*P - Q \quad (20-5-5)$$

$$\text{今令} \quad A - PBR^{-1}B^*P = T \quad (20-5-6)$$

$$-PBR^{-1}B^*P - Q = M \quad (20-5-7)$$

则(20-5-5)式可写成:

$$PT + T^*P = M \quad (20-5-8)$$

这是一个矩阵代数方程。在一般情况下, 可令  $p_{ij}^0 = 0$  ( $i=1, \dots, N, j=1, \dots, N, N$  为系统的阶次) 并代入(20-5-6)、(20-5-7)式求得  $T, M$ , 再将  $T, M$  代入(20-5-8)式, 求得  $P_{ij}^1$ , 经反复迭代, 可求得  $P_{ij}^2, P_{ij}^3, \dots, P_{ij}^l, \dots$  直到满足:

$$\sum_{i,j=1}^N (P_{ij}^{l+1} - P_{ij}^l)^2 \leq E$$

为止。其中  $E$  是预先设定的允许误差。

本程序在求解(20-5-8)式时, 是将该矩阵方程化为  $P_{ij}$  的代数方程组后再对该代数方程组进行计算。如前所述, 由于  $P$  为对称阵, 若系统为  $N$  阶, 则只有  $\frac{N(N+1)}{2}$  个线性无关的  $P_{ij}$  代数方程。

例如, 当  $N=2$  时, (20-5-8) 式为:

$$\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} + \begin{bmatrix} T_{11} & T_{21} \\ T_{12} & T_{22} \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad (20-5-9)$$

化为代数方程组:

$$\begin{bmatrix} 2T_{11} & T_{21} & T_{21} & 0 \\ T_{12} & T_{11}+T_{22} & 0 & T_{21} \\ T_{12} & 0 & T_{11}+T_{22} & T_{21} \\ 0 & T_{12} & T_{12} & 2T_{22} \end{bmatrix} \begin{bmatrix} P_{11} \\ P_{12} \\ P_{21} \\ P_{22} \end{bmatrix} = \begin{bmatrix} M_{11} \\ M_{12} \\ M_{21} \\ M_{22} \end{bmatrix} \quad (20-5-10)$$

考虑到  $P_{ij} = P_{ji}$  ( $i \neq j$ ) 则(20-5-10)式可化为:

$$\begin{bmatrix} 2T_{11} & 2T_{12} & 0 \\ T_{12} & T_{11}+T_{22} & T_{21} \\ 0 & 2T_{12} & 2T_{22} \end{bmatrix} \begin{bmatrix} P_{11} \\ P_{12} \\ P_{22} \end{bmatrix} = \begin{bmatrix} M_{11} \\ M_{12} \\ M_{22} \end{bmatrix} \quad (20-5-11)$$

这样, 就将矩阵方程(20-5-9)式化为(20-5-11)式的代数方程组, 是由  $\frac{N(N+1)}{2} = 3$  个代

数方程组成的。

本程序采用主元素消元法对代数方程组求解得  $P_{ij}$ 。

当然, 不同的  $Q, R$  阵, 则会得到不同的  $P$ , 从而会得到不同的最佳控制, 对设计者来说, 往往需要计算所设计的系统在一定给定信号下的响应, 本程序在求解 Riccate 方程后, 采用四阶龙格库塔法计算系统在阶跃信号作用下系统响应。

关于主元素消元法及四阶龙格库塔法可参阅本书第十四章有关内容。

## 二、程序说明:

### a. 程序中所使用的部分变量说明:

$N$ : 系统状态变量的维数, 即状态方程的阶次。

$M$ : 系统控制变量的维数

$E_2$ : 解Riccate方程迭代次数

$Q_1$ : 计算步距

$Q_2$ : 打印间隔 (即每隔  $Q_2 \cdot Q_1$  打印一点)

$Q_3$ : 打印次数

$J_0$ : 打印变量个数

$J_7, J_8, J_9$ : 打印变量的序号

### b. 数据区说明:

本程序数据区从1000号开始。

为了便于说明, 将系统的结构图附在下面。

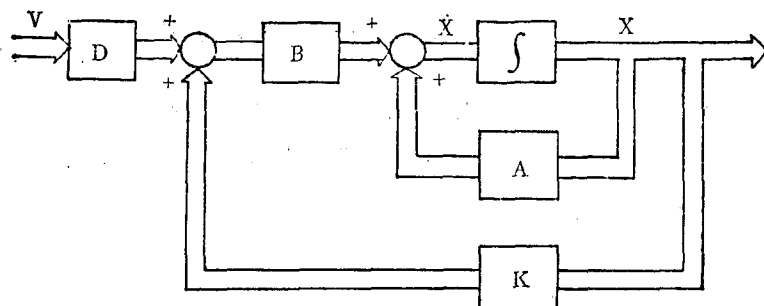


图 20-5-2

系统结构参数:  $A$  ( $N \times N$  阵)、 $B$  ( $N \times M$  阵)、 $D$  ( $M \times M$  阵)

用户给定参数:  $Q$  ( $N \times N$  阵)  $R$  ( $M \times M$  阵)  $P_0$  ( $N \times N$  阵)

$E_1$  (允许误差)

$V$  (用户给定信号  $M \times 1$  阵)

上述参数在数据区按以下顺序安排:

1000	DATA $A_{11}, A_{12}, \dots, A_{N1}$
...	
$1000+N-1$	DATA $A_{N1}, A_{N2}, \dots, A_{NN}$
$1000+N$	DATA $B_{11}, B_{12}, \dots, B_{N1}$
...	
$1000+N+M-1$	DATA $B_{M1}, B_{M2}, \dots, B_{MN}$
$1000+N+M$	DATA $Q_{11}, Q_{22}, \dots, Q_{NN}$
$1000+N+M+1$	DATA $R_{11}, R_{12}, \dots, R_{MM}$
$1000+N+M+2$	DATA $P_{11}^0, P_{12}^0, \dots, P_{1N}^0$
...	
$1000+2N+M+2$	DATA $P_{N1}^0, P_{N2}^0, \dots, P_{NN}^0$
$1000+2N+M+3$	DATA $E_1$

1000+2N+M+4 DATA D<sub>11</sub>, D<sub>12</sub>, .....D<sub>1M</sub>

1000+2N+2M+4 DATA D<sub>M1</sub>, D<sub>M2</sub>, .....D<sub>MM</sub>

1000+2N+2M+5 DATA v<sub>1</sub>, v<sub>2</sub>, .....v<sub>M</sub>

### 三、操作说明:

(1) 由BASIC引入本程序

(2) 按(b)所说明的顺序修改1000号以后的语句, 为运行安排好数据。

(3) 用命令RUN启动本程序运行

此时电传印出"N, M=?", 即询问系统阶次及控制变量的维数。

(4) 用键盘回答N和M的具体值。

计算机即开始执行程序, 首先求解Riccate方程, 每计算一次后进行判断,

若  $\sum_{i=1}^N (P_{ii}^{i+1} - P_{ii}^i)^2 > E_1$ , 则打印累计计算次数, 否则印出P<sub>ii</sub>及K<sub>ii</sub> (i=1, 2, ..., N, J=

1, 2, ..., N, S=1, 2, ..., M) 说明Riccate方程求解完毕。

接着计算机转入计算系统过渡过程的程序。计算机首先印出"Q<sub>1</sub>, Q<sub>2</sub>, Q<sub>3</sub>?", 即询问计算步距, 打印间隔步数及打印次数。

(5) 键盘回答Q<sub>1</sub>, Q<sub>2</sub>, Q<sub>3</sub>之值后计算机又印出"J<sub>0</sub>, J<sub>7</sub>, J<sub>8</sub>, J<sub>9</sub>?", 即询问印出的变量个数以及序号 (本程序最多只能输出三个变量, 若少于三个则可令J<sub>8</sub>, J<sub>9</sub>中的某个或两个均为0)。

(6) 键盘回答J<sub>0</sub>, J<sub>7</sub>, J<sub>8</sub>, J<sub>9</sub>。

以后计算机自动计算, 直到计算完毕。

(7) 如果用户在解 Riccate 方程后不希望立即求解过渡过程, 则可在程序中加一停语句:

442 STOP

即可。用户修改参数后 (如Q, R值) 重新启动。

### 四、例题:

系统状态方程:

$$\begin{pmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \\ \dot{X}_4 \end{pmatrix} = \begin{pmatrix} -8 & -2.176 & 0 & 0 \\ 97.173 & -3.8092 & -2.0954 & 0 \\ 0 & 435.919 & 0 & -871.838 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} + \begin{pmatrix} 460 \\ 0 \\ 0 \\ 0 \end{pmatrix} u$$

$$Q = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.2 \end{pmatrix} \quad R = [1]$$

$$P^0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \quad E = 10^{-18}$$

$$D = [1] \quad V = 10$$

$$N = 4 \quad M = 1$$

$$Q_1 = 0.002 \quad Q_2 = 5 \quad Q_3 = 50$$

$$J_0 = 3 \quad J_1 = 1 \quad J_2 = 2 \quad J_3 = 4$$

# 五、程序清单及运行结果

```

4  PRINT "N,M=";
6  INPUT N,M
8  PRINT
10 LET N2=N*N
12 LET N3=N*(N+1)/2
16 DIM A(N,N),B(N,M),Q(N,N),R(M,M),P(N,N)
17 DIM D(M,N),E(N,N),H(N,N),L(N,N),T(N,N)
18 DIM Z(N3),U(N2,N2),V(N2,N2),W(M,N)
19 DIM X(N3),Y(N3),O(N)
20 DIM S(N2,N2),G(N2,2*N2),F(N2,N2)
21 LET E2=0
30 FOR I=1 TO N
32   FOR J=1 TO N
34     READ A(I,J)
36   NEXT J
38 NEXT I
40 FOR I=1 TO N
42   FOR J=1 TO M
44     READ B(I,J)
46   NEXT J
48 NEXT I
50 FOR I=1 TO N
52   FOR J=1 TO N
54     IF I=J GOTO 62
56     LET Q(I,J)=0
58   NEXT J
60 NEXT I
62 READ Q(I,J)

```

```

64     NEXT J
68  NEXT I
70  FOR I=1 TO M
72     FOR J=1 TO M
74         IF I=J GOTO 82
76         LET R(I,J)=0
78     NEXT J
80  NEXT I
82  READ R(I,J)
84  NEXT J
88  NEXT I
90  FOR I=1 TO N
92     FOR J=1 TO N
94         READ P(I,J)
96     NEXT J
98  NEXT I
99  READ E1
100 FOR I=1 TO M
102     FOR J=1 TO N
104         LET D(I,J)=0
106         FOR K=1 TO M
108             LET D(I,J)=D(I,J)+R(I,K)*B(I,K)
109         NEXT K
110     NEXT J
112 NEXT I
114 FOR I=1 TO N
116     FOR J=1 TO N
118         LET E(I,J)=0
120         FOR K=1 TO M
122             LET E(I,J)=E(I,J)+B(I,K)*D(K,J)
124         NEXT K
126     NEXT J
128 NEXT I
129 LET C5=0
130 PRINT "E2="
131 FOR I=1 TO N
132     FOR J=1 TO N
134         LET H(I,J)=0
136         FOR K=1 TO N
138             LET H(I,J)=H(I,J)+E(I,K)*P(K,J)

```

```

140     NEXT K
142     NEXT J
144 NEXT I
146 FOR I=1 TO N
148     FOR J=1 TO N
150         LET L(I,J)=0
152         FOR K=1 TO N
154             LET L(I,J)=L(I,J)+P(I,K)*H(K,J)
156         NEXT K
158     NEXT J
160 NEXT I
166 FOR I=1 TO N
168     FOR J=1 TO N
170         LET L(I,J)=-L(I,J)-Q(I,J)
172         LET T(I,J)=A(I,J)-H(I,J)
174     NEXT J
176 NEXT I
177 IF C5=1 GOTO 665
180 LET V1=0
182 FOR I=1 TO N
184     FOR J=1 TO N
186         LET V1=V1+1
188         LET Z(V1)=L(I,J)
190         LET X(V1)=P(I,J)
192     NEXT J
194 NEXT I
200 FOR I=1 TO N2
202     FOR J=1 TO N2
204         LET U(I,J)=0
206         LET V(I,J)=0
208     NEXT J
210 NEXT I
220 FOR K=1 TO N
222     FOR I=1 TO N
224         FOR J=1 TO N
226             LET U(I+(K-1)*N,J+(K-1)*N)=T(J,I)
228             LET V(I+(K-1)*N,J+(I-1)*N)=T(J,K)
230         NEXT J
232     NEXT I
234 NEXT K

```



```

240   FOR I=1 TO N2
242     FOR J=1 TO N2
244       LET S(I,J)=U(I,J)+V(I,J)
246     NEXT J
248   NEXT I
250   FOR I=1 TO N2
252     FOR K=1 TO N
254       FOR J=1 TO N
256         LET U(K,J)=S(I,J+(K-1)*N)
258         IF K>J GOTO 452
259       NEXT J
260     NEXT K
261     LET V1=0
262     FOR I1=1 TO N
264       FOR J1=I1 TO N
266         LET V1=V1+1
268         LET V(I,V1)=U(I1,J1)
270       NEXT J1
272     NEXT I1
276   NEXT I
290   LET V1=0
292   FOR K=1 TO N
294     FOR I=1 TO N
296       LET V1=V1+1
298       IF K>I GOTO 460
300       FOR J=1 TO N3
302         LET S(V1,J)=V(I+(K-1)*N,J)
304       NEXT J
306     NEXT I
308   NEXT K
320   GOSUB 700
330   FOR I=1 TO N3
332     LET Y(I)=0
334     FOR J=1 TO N3
336       LET Y(I)=Y(I)+F(I,J)*Z(J)
338     NEXT J
340   NEXT I
350   LET V1=0
352   FOR I=1 TO N
354     FOR J=1 TO N

```

```

356      LET V1=V1+1
358      LET P(I,J)=Y(V1)
360      LET P(J,I)=P(I,J)
362      NEXT J
364      NEXT I
370      LET Y1=0
372      FOR I=1 TO N3
374          LET Y1=Y1+(Y(I)-X(I)) * (Y(I)-X(I))
376      NEXT I
378      IF ABS(Y1)<E1 GOTO 402
379      LET E2=E2+1
380      PRINT E2;
382      GOTO 131
400      PRINT
401      GOTO 420
402      PRINT
403      PRINT "P"
404      FOR I=1 TO N
406          FOR J=1 TO N
408              PRINT P(I,J);
410          NEXT J
412          PRINT
414      NEXT I
420      PRINT "K"
422      FOR I=1 TO M
424          FOR J=1 TO N
426              LET W(I,J)=0
428              FOR K=1 TO N
430                  LET W(I,J)=W(I,J)+D(I,K) * P(K,J)
432              NEXT K
434              PRINT W(I,J);
436          NEXT J
438          PRINT
440      NEXT I
445      LET C5=1
448      GOTO 131
450      END
452      LET U(J,K)=U(K,J)+U(J,K)
454      NEXT J
455      NEXT K

```

```

456 GOTO 261
460 LET V1=V1-1
462 NEXT I
464 NEXT K
466 GOTO 320
500 FOR I=1 TO N
501     LET X(I)=0
502 NEXT I
504 FOR I=1 TO N
505     LET U(I,0)=0
506 NEXT I
507 PRINT "Q1,Q2,Q3"
508 INPUT Q1,Q2,Q3
509 PRINT "J0,J7,J8,J9"
510 INPUT J0,J7,J8,J9
511 PRINT
512 LET Z(1)=0
513 LET Z(2)=Q1/2
514 LET Z(3)=Q1/2
515 LET Z(4)=Q1
516 LET T1=0
517 IF J0=3 GOTO 523
518 IF J0=2 GOTO 521
519 PRINT "T", TAB(12); "X(", J7, ")"
520 GOTO 525
521 PRINT "T", TAB(12); "X(", J7, ")", TAB(24); "X(", J8, ")",
522 GOTO 525
523 PRINT "T", TAB(12); "X(", J7, ")", TAB(24); "X(", J8, ")",
524 PRINT TAB(36); "X(", J9, ")",
525 FOR I1=1 TO Q3
526     FOR J1=1 TO Q2
527         GOSUB 600
528         FOR I=1 TO N
529             LET X(I)=X(I)+Q1*(U(I,1)+2*U(I,2)+2*U(I,3)+U(I,4))/6
530         NEXT I
531         LET T1=T1+Q1
532     NEXT J1
533 IF J0=3 GOTO 540
534 IF J0=2 GOTO 537
535 PRINT T1, TAB(12); X(J7)

```

```

536     GOTO 545
537     PRINT T1, TAB(12), X(J7), TAB(24), X(J8)
538     GOTO 545
540     PRINT T1, TAB(12), X(J7), TAB(24), X(J8), TAB(36), X(J9)
545 NEXT I1
550 END
600 FOR J=1 TO 4
610     FOR I=1 TO N
615         LET V(I,J)=Z(J) * U(I,J-1)
618         LET Y(I)=X(I)
620         LET Y(I)=V(I,J)+Y(I)
625     NEXT I
628     FOR I=1 TO N
630         LET F(I,J)=0
635         FOR L1=1 TO N
640             LET F(I,J)=F(I,J)+T(I,L1) * Y(L1)
645         NEXT L1
646         LET U(I,J)=F(I,J)+Q(I)
650     NEXT I
655 NEXT J
660 RETURN
665 FOR I=1 TO M
666     FOR J=1 TO M
667         READ D(I,J)
668     NEXT J
669 NEXT I
670 FOR I=1 TO M
671     READ O(I)
672 NEXT I
673 FOR I=1 TO M
674     LET V(I)=0
675     FOR J=1 TO M
676         LET V(I)=V(I)+D(I,J) * O(J)
677     NEXT J
678 NEXT I
679 FOR I=1 TO N
680     LET O(I)=0
681     FOR J=1 TO M
682         LET O(I)=O(I)+B(I,J) * V(J)
683     NEXT J

```

```

684 NEXT I
685 GOTO 500
700 REM SUBROUTINE MATINV S(N,N) TO F(N,N)
701 LET N1=N3
702 LET F1=N1+1
704 LET H1=N1+N1
708 FOR I=1 TO N1
710 FOR J=1 TO N1
712 LET G(I,J)=S(I,J)
714 NEXT J
716 NEXT I
718 FOR I=1 TO N1
720 FOR J=F1 TO H1
722 LET G(I,J)=0
724 NEXT J
726 NEXT I
728 FOR I=1 TO N1
730 LET J=I+N1
732 LET G(I,J)=1
734 NEXT I
736 FOR K1=1 TO N1
738 LET C1=K1+1
740 IF K1=N1 GOTO 764
742 LET C2=K1
744 FOR I=C1 TO N1
746 IF ABS (G(I,K1))<=ABS (G(C2,K1)) GOTO 750
748 LET C2=I
750 NEXT I
752 IF C2=K1 GOTO 764
754 FOR J=K1 TO H1
756 LET C3=G(K1,J)
758 LET G(K1,J)=G(C2,J)
760 LET G(C2,J)=C3
762 NEXT J
764 FOR J=C1 TO H1
766 LET G(K1,J)=G(K1,J)/G(K1,K1)
768 NEXT J
770 IF K1=1 GOTO 785
772 LET C4=K1-1
774 FOR I=1 TO C4

```

```

776         FOR J=C1 TO H1
778             LET G(I,J)=G(I,J)-G(I,K1)*G(K1,J)
780         NEXT J
782     NEXT I
784     IF K1=N1 GOTO 790
785     FOR I=C1 TO N1
786         FOR J=C1 TO H1
787             LET G(I,J)=G(I,J)-G(I,K1)*G(K1,J)
788         NEXT J
789     NEXT I
790 NEXT K1
791 FOR I=1 TO N1
792     FOR J=1 TO N1
793         LET K1=J+N1
794         LET F(I,J)=G(I,K1)
795     NEXT J
796 NEXT I
797 RETURN
1000 DATA -8,-2.176, 0, 0, 97.173,-3.8092,-2.0954, 0
1001 DATA 0, 435.919, 0,-871.838, 0, 0, 1, 0
1002 DATA 400, 0, 0, 0
1003 DATA 0, 0, 0, .2
1004 DATA 1
1005 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
1006 DATA 1E-15
1007 DATA 1
1008 DATA 10

```

#### 运行结果

RUN

N,M=? 4? 1

E2=

1 2 3 4 5 6 7 8 9 10 11

P

```

6.2805E-4 3.76444E-4 2.30416E-5 2.85219E-4
3.76444E-4 3.45735E-4 3.0907E-5 5.2516E-4
2.30416E-5 3.0907E-5 4.02593E-6 1.07235E-4
2.85219E-4 5.2516E-4 1.07235E-4 5.66188E-3

```

K

```

.25122 .150578 9.21663E-3 .114088

```

Q1, Q2, Q3

? 0.002? 5? 50

J0, J7, J8, J9

? 3? 1? 2? 4

T	X(1)	X(2)	X(4)
.01	21.9311	12.9963	5.58064E-2
.02	20.6008	32.6344	.689994
.03	11.1079	42.6817	2.63959
3.99999E-2	2.65273	40.066	6.16057
4.99999E-2	-.863508	29.3221	10.8524
5.99999E-2	.582126	17.4523	15.87
6.99999E-2	4.88842	10.2261	20.2868
7.99998E-2	9.43964	10.3362	23.4174
8.99998E-2	12.2362	17.2528	24.9935
9.99997E-2	12.417	28.2227	25.1679
.11	10.2356	39.7255	24.3871
.12	6.67961	48.7844	23.2057
.13	2.9624	53.761	22.1118
.139999	8.32493E-2	54.5284	21.4152
.149999	-1.42776	52.1404	21.2133
.159999	-1.55322	48.2304	21.4263
.169999	-.665126	44.3954	21.8724
.179999	.682809	41.7509	22.3511
.189999	1.96213	40.7452	22.7081
.199999	2.81445	41.2163	22.8684
.209998	3.10784	42.6109	22.8364
.219998	2.91203	44.2567	22.6728
.229998	2.4217	45.5937	22.4609
.239998	1.86441	46.3083	22.2758
.249998	1.42439	46.3581	22.1643
.259998	1.20022	45.9093	22.1387
.269998	1.19937	45.2299	22.183
.279998	1.36119	44.5829	22.2648
.289997	1.59355	44.1496	22.3496
.299997	1.80874	43.9987	22.4118
.309997	1.94816	44.0971	22.4391
.319997	1.9916	44.347	22.4332
.329997	1.95267	44.6333	22.4049
.339997	1.86548	44.8615	22.3688
.349997	1.76901	44.9804	22.3377

.359997	1.69422	44.9851	22.3193
.369996	1.65718	44.9056	22.3156
.379996	1.65843	44.7885	22.3236
.389996	1.68705	44.6787	22.3378
.399996	1.72708	44.6065	22.3523
.409996	1.76358	44.5829	22.3627
.419996	1.78676	44.6015	22.3671
.429996	1.79345	44.6451	22.3658
.439996	1.78622	44.6941	22.3608
.449995	1.77102	44.7325	22.3546
.459995	1.75453	44.752	22.3493
.469995	1.74196	44.7519	22.3463
.479995	1.73593	44.7377	22.3458
.489995	1.73644	44.7175	22.3472
.499995	1.74152	44.6988	22.3497

##### 五、几点说明:

1. Q、R阵是由用户给定的,一般是取对角线阵。否则,在数据区安排Q、R时用户必须参照A阵的顺序安排Q、R的数据。

2. 关于P阵的初值问题,这也是由用户给定的。若系统中不存在纯积分环节(即A阵中某一行只有一个元素不为零,而该元素所在列的其它元素不全为零)的情况下,可设P的初值为零阵。否则P<sup>0</sup>不能取零阵,此时为了求得P之初值,可先将该积分环节变成大惯性环节(即在该行增加一个常数),用本程序求得P<sup>1</sup>,再以此P<sup>1</sup>做为原系统 Riccate 方程的初值即可。

##### 本章参考文献

- 〔1〕 郑大钟编  
《现代控制理论》 1979年,清华大学自动化系
- 〔2〕 熊光楞编  
《控制系统的数字仿真》 1980,清华大学自动化系





## 一、基本语句

LET  
INPUT  
READ  
DATA  
RESTORE  
GOTO  
IF  
GOSUB/RETURN  
FOR  
NEXT  
PRINT  
DIM  
REM  
DEF  
STOP  
END

赋值语句  
键盘输入语句  
读语句  
数语句  
恢复数据区指针  
无条件转问语句  
条件语句  
转子与返回语句  
循环初始语句  
循环终止语句  
打印输出语句  
维数说明语句  
注释语句  
定义函数语句  
暂停语句  
结束语句

## 二、键盘命令

RUN  
LIST  
NEW

运行程序  
列清单  
清除工作区

## 三、标准函数

SIN(X)  
COS(X)  
TAN(X)  
ATN(X)  
LOG(X)  
EXP(X)  
SQR(X)  
ABS(X)  
INT(X)  
RND(X)  
SGN(X)

正弦函数  
余弦函数  
正切函数  
反正切函数  
对数函数 (自然对数)  
指数函数  
平方根函数  
绝对值函数  
取整函数  
随机函数  
符号函数